Contextual Injection of Quality Measures into Software Engineering Processes

Gregor Grambow and Roy Oberhauser
Computer Science Dept.
Aalen University, Germany
{gregor.grambow, roy.oberhauser}@htw-aalen.de

Manfred Reichert
Institute for Databases and Information Systems
Ulm University, Germany
manfred.reichert@uni-ulm.de

***Abstract*—Despite improvements in software engineering processes and tools, concrete preventative and analytical software quality assurance activities are still typically manually triggered and determined, resulting in missed or untimely quality opportunities and increased project overhead. Quality goals, when defined, lack holistic environmental support for automated performance measurement and governance that is tightly integrated in the low-level operational software engineering processes. This results in higher quality risks and cost risks. Based on adaptive process management, an approach is presented that injects situationally-determined quality measure proposals into the concrete workflows of software engineers, using contextual semantic knowledge and multi-agent quality goal tracking and decision-making. Our evaluation shows the feasibility of the approach for automatically providing timely quality measure guidance to software engineers without disrupting their current activities. This supports process governance while reducing quality risks and costs during software development projects.**

Keywords-software quality assurance; process-centered software engineering; adaptive process management; semantic technology; agents; Goal-Question-Metric technique

I. INTRODUCTION

This article extends our previous work in [1], which described aspects of an approach for automated integration of software quality management and software engineering process management. Today IT-supported business process management (BPM) enjoys wide industrial adoption [2] and can support improved product quality by ensuring that quality-supporting processes are executed [3]. Process repetition and predictability lowers the recovery risk for necessary investments in process modeling, process management system support, and enterprise application integration [5]. Interestingly, BPM is also increasingly being used for product development [4].

In the software engineering (SE) domain, numerous obstacles inhibit automated SE process management (SEPM) at the operational level. These include the contextual dependency of the low-level activities (e.g., coding, debugging, testing), the high degree of change to the involved artifacts (e.g., source code files, test documentations), the informational and environmental dependencies (e.g., coordination, requirements, reports, tools), the uniqueness of each developer's concrete personal process (e.g., junior vs. senior engineers, information needs), activity coordination with the overall team process, the contextual and project influences on the processes (e.g., schedule, resource availability), and software quality assurance (SQA) dependencies (e.g., quality plan, reactive quality measures, metric dependencies).

Historically software development projects have also faced difficulties in meeting budget, schedule, functionality, and quality targets [6][7][8]. A more recent study in 2002 by the National Institute of Standards and Technology (NIST) found that most delivered software products are still stricken by bugs and defects [9]. While some of these difficulties might be ascribed to a misaligned planning environment in certain organizations [10], the project pressures and resulting issues will likely linger due to global competition and other influences [11]. Other difficulties can be attributed to SE's adolescence as a discipline and certain unique product properties that affect the SE development process, such as software's complexity, conformity, changeability, and invisibility [6]. Additionally, the extent (too little or too much) and timeliness of SQA significantly impacts overall project costs [12], making effective and efficient SQA vital. Yet it remains laborious to manage and apply the appropriate low-level SQA measures (actions) in a timely fashion during SE process enactment. In order to achieve software quality goals, these must be defined and concretely and contemporaneously measured [13]; yet this is often challenging for various SE organizations [14]. Especially small and medium sized companies often struggle to achieve high quality levels. This often results from the increased complexity of their growing organizational structures, the lack of process maturity and capabilities, and the lack of dedicated quality management personnel.

A. Problem Statement

While SE process models foster development efficiency [15], they are often defined rather abstractly and thus fail to provide low-level guidance for the activities actually executed at the operational level. Furthermore, processes are often defined rigidly beforehand. However, during their execution, reality often diverges from the planned process [16].

Automated guidance for combining SQA with SEPM is not yet prevalent. Challenges in software development projects are presented at both the product and process levels based on the nature of software artifacts and manually driven processes. Product intangibility hinders effective retrieval of

timely information about its product quality status. Additionally, the combination of abstract process definitions and intertwined, contextually information-dependent lower-level workflows make targeted process guidance for developers irrelevant, complex, or financially infeasible. Thus, artifact issues often cannot be detected promptly and, even if they are, the contemporary integration of quality measures into the workflow is not possible. At times, quality measures come into focus and are applied close to release, or, when the project is behind schedule, they may be jettisoned altogether; however, it is generally acknowledged that their application in earlier development stages saves time as well as money [12][17]. The proper application of quality measures is also problematic, since their effectiveness and efficiency depend on many factors, such as the applicability of the measure, the project timing, worker competency, and correct fulfillment [12]. For clarity, measure in this article is meant in the sense of a specific action intended to produce some effect - reactive actions are thus countermeasures.

To illustrate the problem as well as the proposed solutions, this paper uses a series of simplified practical scenarios dealing with a fictional company (called 'the Company').

Example 1 (Abstract Process): *The Company uses a SE process model for software development, the V-Model XT [18]. This process features detailed descriptions of activities, roles, milestones, and artifacts. Its application is based on the use of various documents with no automated governance or support. In the Company, activities for developers are planned and scheduled on a very coarse-grained level, leaving the coordination of what is to be done to the developers and manager. Therefore, the SE process does not really "touch" the developers, and their actual activities are difficult to trace. The quality of the source code is not monitored continuously and static code analysis tools are only used sparsely by the developers. Deterioration of the source code quality goes undetected and quality measures are only taken at the end of projects when there is time left or when concrete bugs exist.*

B. Contribution

Automated support for and governance of the coordinated integration of SQA in SEPM offers promising perspectives for addressing shortcomings in current SQA approaches. In the following, the terms *process* and *workflow* will be used extensively, and are delimited here against each other in alignment with existing definitions provided by the Workflow Management Coalition [19] and Gartner Research [20].

Definition: Business Process Management deals with the explicit identification, implementation, and governance of processes as well as their improvement and documentation. This incorporates different issues such as organizational or business aspects, or the strategic alignment of the activities. Workflow Management, in turn, deals with the automation of business processes. Hence, a workflow is the technical implementation of a business process or a part of it.

In our previous work, we introduced the CoSEEEK framework [21], which utilizes various technologies to provide automated, context-aware assistance for SEPM. In [22] and [23], we provided a solution to dynamically generate workflows according to the properties of various situations to support dynamic workflows extraneous to the SE process. We are currently also working on the integration of this dynamic generation with workflows belonging to the SE process and covering situations where pre-planned workflows are too rigid. In [24], we introduced an SE workflow language that provides extended modeling capabilities for SE workflows and improves the connection between abstractly defined processes and concretely executed activities. Finally, in [1] and [25], which provide the basis for this article, we described aspects of our overall approach for integrating SQM and SEPM. In particular, this paper provides a more comprehensive description of this approach with further extensions, in particular elucidating the following areas:

- automatic detection and management of source code related problems in a SE project,
- automatic assignment of quality measures to detected quality problems,
- automatic strategic prioritization and alignment of quality measures to project quality goals,
- tailoring of measure (action) proposals to the situation, and
- automatic integration of quality measures in the software engineer's workflow.

The remainder of this paper is organized as follows: Section II presents background information needed for the understanding of this paper and elicits fundamental requirements for our solution approach. Section III describes our solution approach. Section IV discusses realization aspects and Section V evaluates our solution. Finally, Section VI discusses related work and Section VII concludes the paper.

II. REQUIREMENTS

Requirements have been elicited based on various sources we found in literature. The identified requirements cover different areas to enable comprehensive system support for integrating SQA and SEPM.

Context-awareness. To enable automated decisions on quality measure assignments, any system support should be aware of its environment and the context of the current situation.

Requirement R:Ctx1 (Context integration): To be aware of problems in the SE project, the system must have a facility to integrate information on SE process or product problems from various sources (e.g., external tools measuring the state of the source code, bug tracking systems).

Requirement R:Ctx2 (Quality opportunity awareness): To enable automated integration of quality measures at run-time, the system must be aware of quality opportunities, meaning time points when a user can cope with a quality

measure. This requires knowledge about the users' schedule, meaning the abstract activities that have been scheduled and estimated for the user.

Process management: To enable the automated integration of quality measures, the system must be able to govern the SE process automatically. To foster this in SE, facilities must be in place to enable the system to match the workflow specification belonging to the process (or parts of it) with other facts representing the current situation. That way, contextual information can be used for SE process support.

Requirement R:Sepm1 (Vertical workflow connection): It should be possible to define flexible connections between different workflows (meaning the vertical connections between sub-workflows and their super-workflows). In traditional process management, a simple connection between sub- and super-workflow is possible. If an activity of a super-workflow refers to a sub-workflow, it will be only finished at run-time after completing the corresponding sub-workflow instance. However, in practice, more complex connections may be required as illustrated in Figure 1 and confirmed by processes from other domains like the automotive industry [26] and enterprise resource planning [27]. In this example (Figure 1), activities are grouped by work packages. In the planning phase, for example, the packages and their corresponding activities are planned. This means that the activity of planning a package depends on the completion of the planning of the contained activities. The same applies for the processing of a package. That way there are multiple connections between the super and the sub-workflow, and the completion of a certain activity does not necessarily depend on the completion of a whole sub-workflow, but on the completion of one or multiple activities in one or multiple other workflows.

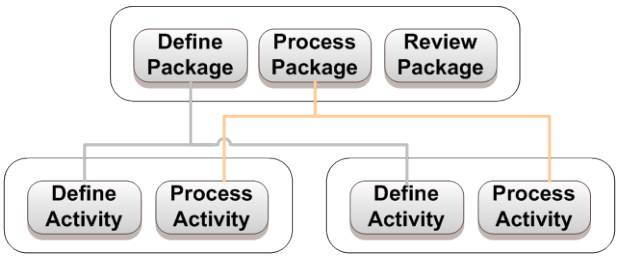

Figure 1. Sub-workflow Connections.

Requirement R:Sepm2 (Task Granularities): For the automated detection of quality opportunities, an explicit connection between abstract assignments, which have been planned and scheduled, and the related concretely executed activities is desirable. Traditional BPM features human tasks only with one single granularity. In fact, human tasks exist on different levels of abstraction that are often related to each other (see also [28][29]). Process management lacks sufficient support for this - just modeling tasks on different levels of abstraction does not adequately match reality since tasks usually have different properties. An example of those tasks is shown in Figure 2.

Example 2 (Task Granularities): *Task 1 is an abstract assignment, which was planned and estimated from the business side. That task implies Tasks 2 and 3, which are concretely planned and executed by the developer. Those tasks, in turn, also imply tasks on a more concrete level like Tasks 4, 5 and 6, which may have special connections to the environment, as they require, for example, certain tools.*

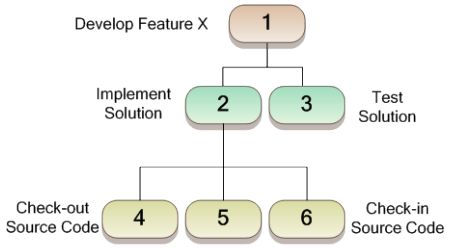

Figure 2. Task Granularities.

Requirement R:Sepm3 (Workflow adaptation): The concrete workflows should be adaptable. In particular, their specification should support automated adaptations during run-time to enable the system to automatically and dynamically insert quality activities into workflows where required and favorable.

Quality Measure Selection. The selection of appropriate quality measures during SE process execution constitutes another challenge. Various factors must be taken into account for effectiveness and efficiency.

Requirement R:Qmsel1 (Quality measure selection): Applied quality measures should be automatically chosen during run-time in alignment to project goals in order to match the defined strategy of the project.

Requirement R:Qmsel2 (Proactive measures): Quality measures should not only rely on detected problems, but also consider common quality enhancement. Thus, proactive and reactive measures should be available.

Requirement R:Qmsel3 (Situational measure tailoring): Context-sensitive tailoring of proposed measures is desirable considering different factors of the actual situation, e.g., properties of the applying person and application time point.

Requirement R:Qmsel4 (Measure assessment): The selection of measures should be aware of their effectiveness to optimally match with specific environments or situations in different companies. Therefore, continuous monitoring of the quality of the source code is essential to detect potential impacts of applied measures on the overall quality. In particular, a relation between the application of SQA measures and the evolution of source code quality should be established to assess the effectiveness of the measures.

III. SOLUTION APPROACH

Considering the aforementioned requirements, the concepts behind our solution approach are now described in detail. To automatically integrate quality measures into the SE process, our approach consists of a process (referred to here as a procedure to avoid confusion with SEPM) in conjunction with the architecture of the *Context-aware Software*

Engineering Environment Event-driven frameworK (CoSEEEK) [21].

A. Solution Procedure

Our solution procedure involves three fundamental phases: a *detection phase*, a *processing phase*, and a *post-processing phase* as shown in Figure 3. These phases as well as their different steps will be explained in the following subsections.

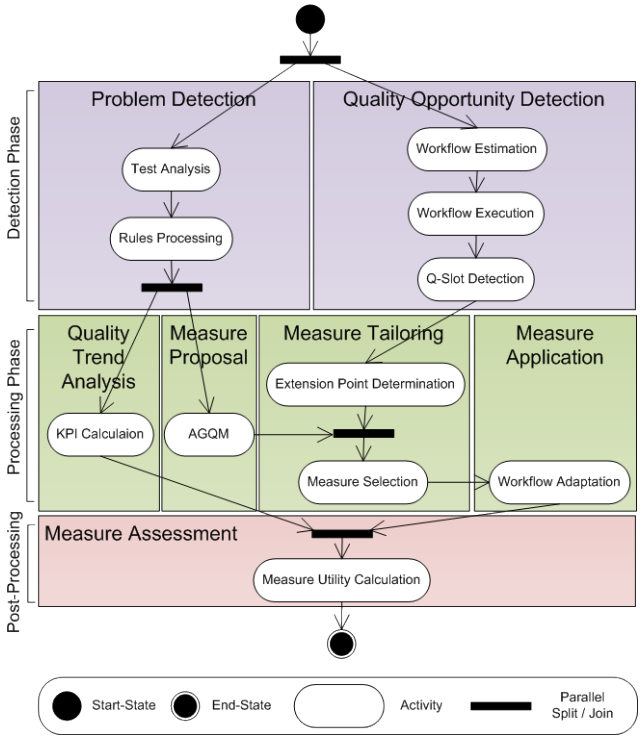

Figure 3. Conceptual Procedure.

The *Detection Phase* continuously enables an awareness of the current project situation to meet the requirements relating to context-awareness (cf. Section II.B). For integrating quality measures, two factors are of particular interest: the presence of problems (cf. Requirement R:Ctx1) - recognized via the 'Problem Detection' - and the availability of opportunities for quality measures in the users' schedule (cf. Requirement R:Ctx2) - recognized via 'Quality Opportunity Detection'. To enable such detection in an automated fashion, the SE process specification must be extended (cf. Requirement R:Sepm2). The applied extensions will be described in Section III.C.

The *Processing Phase* deals with the selection and proposal of the quality measures and involves four steps. Utilizing the GQM technique [30], quality measures (actions) are initially proposed in alignment with project goals to satisfy Requirement R:Qmsel1. This phase also adds proactive measures to the measure proposal process (cf. Requirement R:Qmsel2). To prepare these measures for their automated application, 'Measure Tailoring' incorporates information about the applying persons and the possible points in the users' schedule in which to apply the measure (cf. Requirement R:Qmsel3). This leads to a selection of appropriate points (so called *Q-Slots*) and to an automated integration of the quality measures into the concrete workflow of the chosen person. The application of measures can be also done automatically utilizing extensions made to the SE process specifications (cf. Requirement R:Sepm3). These enable the system to be aware of matching extension points (e.g., in the workflows). These are illustrated in Figure 4 by a small abstract workflow containing the activities 'A1' to 'A5'. 'A2' and 'A4' have an associated extension point, meaning that an automated insertion of a new activity is possible subsequent to these activities.

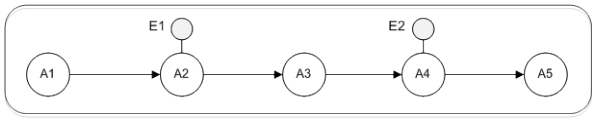

Figure 4. Extension Points.

Finally, to be able to track the quality of the project continuously, in the *Post-Processing Phase* (cf. Figure 3) a 'Measure Assessment' is performed via a quality trend analysis. This analysis supports an awareness and automatic assessment of the potential utility of the applied measures, fostering quality (cf. Requirement R:Qmsel4).

Since each project is unique, the applicability and effectiveness of measures can vary with respect to different projects. Therefore, the system executes an assessment phase to rate the applied measures and to incorporate their impact in the given project.

B. Conceptual Architecture

CoSEEEK provides the necessary infrastructure for realizing the solution procedure presented in the previous sub-section. Its conceptual architecture is shown in Figure 5.

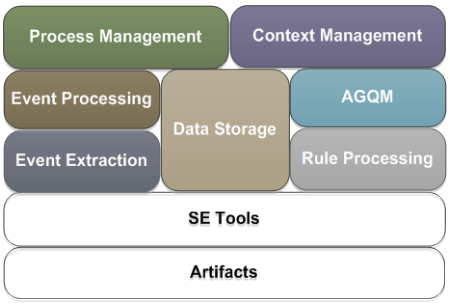

Figure 5. CoSEEEK Conceptual Architecture.

SE Tools is a placeholder for all tools used in a SE project of which CoSEEEK is aware, such as source control systems or IDEs. Artifacts are those things produced in a SE project using the *SE Tools*. This includes source code artifacts, documents, and models.

Awareness of changes to the state of tools as well as artifacts is supported by the *Event Extraction* module. It utilizes sensors that are typically integrated into the tools or otherwise monitor the tools. These sensors generate events in in response to various situations (e.g., 'switch to debug

perspective' in an IDE). The *Data Storage* module encapsulates the storage mechanisms for events and shared data. Communication is event-based and loosely coupled to support integration and exchangeability of different modules.

The events generated and collected in the *Event Extraction* module are basic and low-level. The *Event Processing* module utilizes *complex event processing* (CEP) [31] to process these events, providing high-level events with enriched semantic value. The *Rules Processing* module uses rule-based computing to analyze tool data such as static analysis reports or metrics, and it triggers follow-up actions as necessary (e.g., quality measures for violated metrics). These triggered measures are subsequently filtered by the *AGQM* (Automated Goal Question Metric) module, which automates and extends the GQM approach via multi-agent computing to analyze the project quality state in alignment with strategic project goals and to propose appropriate proactive and reactive quality measures.

The *Process Management* module applies dynamic and adaptive process management technology to govern the activities of the involved project participants. This enables the system to match workflows to real project situations (instead of rigidly prescribing certain activities and their orders) and thus provides real situational guidance. This becomes possible by utilizing the cumulated knowledge contained in the *Context Management* module. In that module, high-level information of all project areas is collected, as, for example, the skills of users or information about the quality state of the project. Using semantic technology, this information can be used to reason about the project state and contextual influences (causes and effects) and thus provide automated decisions and workflow adaptations such as the automated and dynamic integration of quality measure activities during SE process execution.

C. *Context-aware Business Process Management*

To support a high degree of automated and context-aware assistance, a tight coupling of the *Context Management* and the *Process Management* module is required, which will be referred to as *Context-aware Process Management* (CPM). This addresses many of the shortcomings of traditional BPM (as listed in Requirements *R:Sepm1 to R:Sepm3*) and facilitates the comprehensive utilization of the awareness capabilities in CoSEEEK. Fundamentally, process management concepts are enhanced with semantic information. This additional information is stored in the *Context Management* module, while the workflows are managed by the *Process Management* module. Since *Context Management* unifies all project knowledge, it can be also used as a management layer around the *Process Management* module, facilitating context-based process management. Thus, all process-related actions are addressed by the *Context Management* module, which, in turn, manages the actions of the *Process Management* module. Figure 6 illustrates these extensions to process management.

The *Process Management* module governs the workflows and their activities. These two concepts are mirrored in the *Context Management* module: the activity by the *Work Unit* and the workflow by the *Work Unit Container*. Thus, process management is separated into two areas that we call *vertical* and *horizontal process management*. *Horizontal process management* denotes the governing of the different activities of one workflow (also denoted as process orchestration) utilizing well-established workflow patterns like AND, SPLIT, or LOOP. This is done within the *Process Management* module. *Vertical process management,* in turn, deals with the management of the dependencies between different workflows on different levels of abstraction. Since process management only offers one kind of connection (an activity depends on a sub-workflow) here, this is handled by the *Context Management* module. This allows for the flexible definition of dependencies. The completion of a *Work Unit* can depend upon one or multiple *Work Unit Containers* or on one or multiple *Work Units* contained in other *Work Unit Containers*. A mixture of both is possible as well.

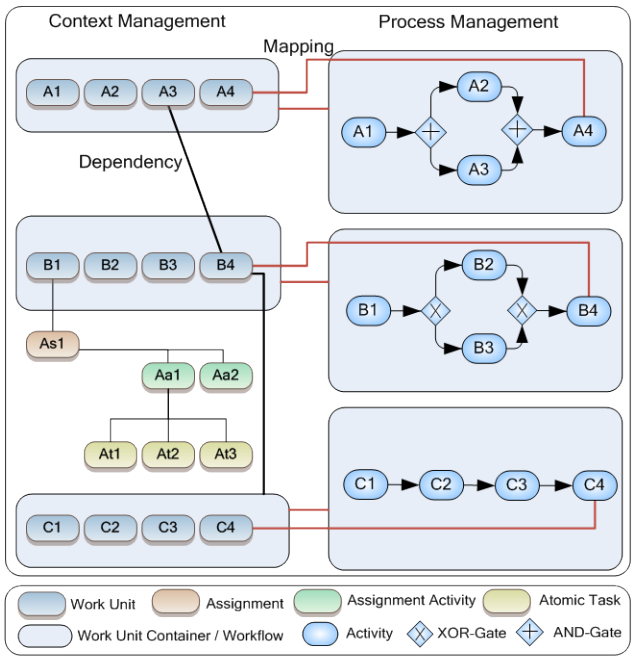

Figure 6. Context-aware Business Process Management.

Work Units are connected to three other concepts, enabling advanced task management (cf. Requirement R:Sepm1). The *Assignment* is used as a coarse-grained top-level task, which is also estimated and scheduled from the business side in a project, exemplified in Figure 2 as "Develop feature X". The *Assignment Activity* then describes the tasks that are necessary to accomplish the *Assignment*, e.g., "Design Solution" or "Write Developer Tests" (cf. Figure 2). The most fine-grained level is described by atomic tasks like "Check out" or "Build Code".

Combining the *Context Management* and the *Process Management* modules enables the automatic adaptation of running workflows based on the current context. This has been used to automatically build workflows for issues extraneous to SE process models, like bug fixing or refactoring as described in [22].

D. *Quality Opportunity Detection*

To enable the automated detection of quality opportunities, an awareness of the activities that have been planned and scheduled becomes necessary. These are captured by the *Assignment*, which has certain properties to capture estimated durations. These assignments can be created, estimated, and scheduled in CoSEEEK or imported from other tools. This is illustrated by Figure 7.

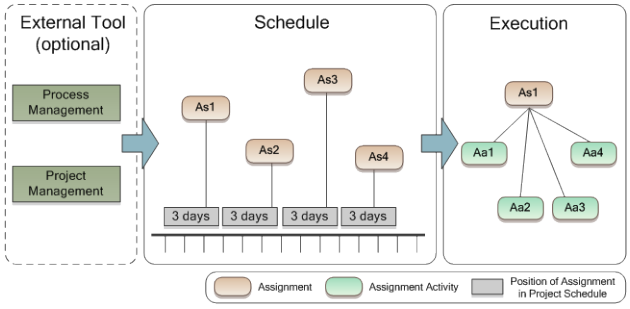

Figure 7. Quality Opportunity Detection.

The figure shows a simple example schedule containing the Assignments 'As1' – 'As4' that are estimated to take three days each. The scheduled Assignments are then taken for execution. The figure illustrates the connection of the Assignment 'As1' to four Assignment Activities for execution, which are 'Aa1' – 'Aa4'. Optionally, the schedule can be imported from an external tool. That way, the activities can be estimated and scheduled from the business side, e.g., utilizing a process management tool like microTOOL in-Step [32], and then be automatically used for execution in CoSEEEK.

In our approach, two triggers for a quality opportunity are implemented. The first one is early *assignment* completion. If a user finishes an assignment earlier than necessary, a quality measure can be assigned to him without delaying forthcoming activities. The second trigger is the quality overhead factor. It enables the *a-priori* specification of a certain percentage of the project workload that should be reserved for quality activities. If the user has not yet reached that amount during process execution, a quality measure may be applied. This can be combined with a quality function indicating how much time for quality should be spent in which project phase. Since it has been shown that it can be beneficial to adjust the work allocation for quality based on the stage of a project [12][17], this can improve both the effectiveness and efficiency of the quality efforts taken.

Example 3 (Quality Opportunity Detection): *To illustrate how automated quality opportunity detection could work, Figure 8 shows a simple schedule of the Company. This example, which demonstrates early assignment completion, assumes that the Company has already adapted the planning to create more fine-grained assignments not taking weeks but days. The schedule comprises four users having five assignments each. Each of these assignments, in turn, is estimated to take three days. Every user in this example finishes early on one Assignment, triggering the creation of a Q-Slot filling the hole in the schedule.*

Figure 8. Schedule with Q-Slots.

The concrete detection of the quality opportunities is done each time an assignment completes. This can be detected automatically by connecting the *Assignments* to the users' *Atomic Tasks* (cf. Section III.C). *Atomic Tasks,* in turn, are connected to the development environment via the sensor infrastructure provided by CoSEEEK, generating an awareness of their status. When all *Atomic Tasks* of an *Assignment Activity* are completed, the *Assignment Activity* completes as well. The same applies to the *Assignments* in relation to the *Assignment Activities*. This process is further described in [25].

E. *Problem Detection*

Problem detection makes use of the environmental awareness of CoSEEEK to identify potential problems, e.g., in the source code. In this context, external data from tools needs to be integrated. This information is utilized for calculating various metrics that can be customized to measure the quality state of the SE project. Metrics directly indicating problems in the source code are obtained from static code analysis tools like PMD [33] and FindBugs [34], while certain testing problems can be detected with test coverage tools [35] such as Cobertura [36] or EMMA [37].

However, not only code-related product-level problems threaten quality, but process-related factors should also be assessed to ensure quality. These assessments include functional testing, profiling, and load testing. Since CoSEEEK is aware of the execution of respective activities, it can ascertain their absence. Thus, process metrics can also include these facts. Facts available to CoSEEEK can be incorporated in metrics, which enables quality awareness through the presence of measurable, quantifiable information. To reduce the associated configuration effort, a set of pre-configured default metrics will be included with the system.

After detecting any problem, measures (actions) can be used to counter them. The *Rules Processing* module is utilized for triggering the automatic proposal of a measure when the threshold for a particular metric has been violated. Metric or violation reports are received and analyzed, and a list of the violated metrics including assigned quality measures is created by the module.

Example 4 (Source Code Monitoring): *Company policy includes a nightly built process on a build server that invokes static code analysis tools to enable continuous measurement. Thus, a deterioration of the source code quality can be detected by metrics such as its cyclomatic complexity. If complexity exceeds a threshold, the code becomes difficult to maintain and test, and there is a higher probability of introducing defects.*

F. Measure proposal

At this point in the procedure, problems as well as Q-Slots have been detected and an initial assignment of quality measures to metric violations has been done. The generation of a Q-Slot then triggers a measure selection and proposal process. The latter is coordinated by the *Context Management* module. First, the *Context Management* module triggers the *AGQM* module to prioritize the measures strategically in alignment to the quality goals of the SE project. Thereafter, the measures are tailored to the current situation.

The process of prioritizing measures is very dynamic due to different goals, presence of various metrics and violations and different project situations. Therefore, the *AGQM* module features a multi agent system (MAS) to prioritize measures in alignment to the project goals to be able to accommodate these various factors. The process is based on an extension to the GQM technique [30]. GQM consists of a hierarchical structure that starts with the definition of certain project goals. During configuration, for each goal various questions are defined. These answers should provide indicators for the level of goal fulfillment. Each question can be associated with certain metrics, establishing a connection from the abstract project goals to concrete facts in the project. The following example shows the application of the GQM technique.

Example 5 (GQM Plan): *As part of a GQM plan, a goal 'Maintainability' could be defined relating to the maintainability of the produced source code. For this goal, one question could be 'How understandable is the code?' For this question, in turn, different metrics could apply. One example is the metric 'Comment Ratio'.*

1) GQM Extensions

This subsection shows the basic concept on which the agents rely. Two main requirements have to be satisfied to facilitate automatic support for GQM execution. First, a GQM plan must exist that defines the relations between goals, questions, and metrics. Second, the metrics have to be integrated in the system, enabling the automatic extraction of corresponding values and thus the automatic receipt of possible deviation information.

Some extensions to the GQM technique became necessary to support automation. Different abstractions of key performance indicators (KPIs) were introduced to enable the automatic calculation of goal deviations. Furthermore, metrics are encapsulated in KPIs to enable consolidation and simplified deviation calculation. Since multiple metrics may be utilized for a single question in GQM, a QKPI (Question KPI) was created for consolidation of the metric values at the question level. Similarly, multiple questions may apply to a single goal, thus a GKPI (Goal KPI) is used for goal deviation calculations. For each of the KPIs, formulas specify how metrics are combined. To support automated multiple goal attainment, each defined goal was assigned one agent responsible for monitoring and fulfillment of that goal.

The calculation of the different KPIs is conducted by the Context Management module as part of the quality trend analysis described in Section I. Figure 9 shows the relation between the different conceptual elements.

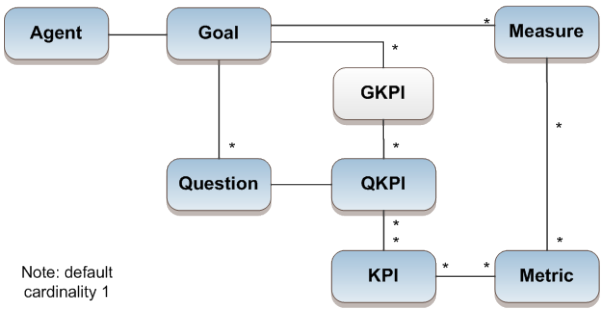

Figure 9. Extended GQM Structure

To prescribe appropriate countermeasures for (potential) quality deterioration, measures were categorized as follows: *reactive (or analytical) measures*, which are directly associated with concrete metrics or violations, and *proactive (or preventative) measures*, which hereby are categorized as supporting certain quality goals at an abstract level and may not be readily associated with a concrete problem. Proactive measures are assigned to GKPIs and can be triggered either when a GKPI deviation occurs (a supportive role) or in the absence of reactive measures. This differentiation is pragmatic since reactive measures can be based on concrete existing problems and can thus be more fine-grained, whereas proactive measures support a goal in general.

2) AGQM Process

At the beginning of a project or a phase or iteration, a quality manager assigns points to each goal (implying its importance) and chooses a bidding strategy for the agent managing that goal. These points are used by agents for negotiating proposed measures. The AGQM process invokes a proactive as well as reactive selection mechanism that results in a measure proposal.

Quality goals can be conflicting, and determining the appropriate balance is project-specific. Thus, a competitive bidding process among agents is chosen for enabling proactive measures, whereas a cooperative voting process is applied for enabling reactive measures. The competitive bidding allows agents with greater importance to definitively have opportunities to support their goal with measures, in contrast to voting where agent majorities might win. That way, a group of lower-priority goal agents does not hinder a higher priority goal from ever asserting influence. The bidding strategies enable agents to win opportunities earlier or later in an iteration cycle.

The reactive voting process is cooperative since a potentially large number of concrete reactive measures based on metric violations become possible for a limited number of quality opportunity slots (Q-slots), and those measures that will have the greatest overall quality impact across all goals are favored. The agents cooperatively vote on the measures list received from the *Rule Processing* module. Via the structure shown in Figure 9, each agent determines for each measure whether a measure belongs to a metric being related to the agent's goal. The points of an agent are then

distributed (currently uniformly) across all measures associated to its goal. A prioritized list of reactive measures is output, while the ultimate choice will be applied by *Context Management* based on the situation.

The proactive section of the *AGQM* module utilizes metrics for calculating the different KPIs, QKPIs, and GKPIs. If there are any deviations at the goal level of an agent (GKPI), it may participate in the bidding session (this favors those goals known to be at risk). Each agent bids and the highest bid wins, elevating its proactive measure set to a proposal. In this process, not just the points differentiate between the goals, but also the strategy chosen by the agents. The strategies influence how an agent increases or decreases its bids after winning or losing for the next bidding process. Choosing a defensive strategy for an agent will increase the likelihood that a proposal of its associated measures will occur in later phases of the iteration. This behavior occurs because in early sessions the agents with more aggressive strategies will place much higher bids. The defensive agent can then place winning bids later when the aggressive agents run out of points.

To define an appropriate proportion between proactive and reactive measures, a proactive-to-reactive ratio can be defined. This determines how often reactive vs. proactive measure sets are provided by the *AGQM* module. Section V will evaluate a concrete scenario utilizing this approach. If no metrics and thus no reactive measures are yet available, then no question or goal deviation is detectable since there is no basis for their calculation. In this case, all agents participate in proactive bidding so that any Q-Slots can be used for proactive measures.

G. Measure Tailoring

The AGQM module has created a list of prioritized measures according to project goals. However, the final selected measure should depend on environmental factors for the most effective and efficient measure application. These include properties of the measure itself, properties of the applying person, and properties of the current situation.

The properties of the measure are defined in the *Context Management* module and include, for example, the type of the measure or the applicable number of users involved (e.g., a code review involves multiple persons). One property of the person that can be of interest is the skill level. The properties of the current situation are modeled based on the concepts of the *Q-Slot* and the *Extension Point*. The *Extension Point*, as part of the semantic enhancements to the process, is a pre-specified point in the process where the integration of a quality measure is feasible, as illustrated in Figure 4. That involves an abstraction level and applicable measure type since, for example, at the end of a project phase other measures might be applicable compared to a time point after directly implementing new functionality. The *Q-Slot* captures a time category indicating how much time is left for a quality measure. Via these properties, a measure fitting to the current situation can be chosen. This process is further explained in [25].

H. Measure Application

To enable a high degree of automated guidance, the user is not only informed about the measure to be applied, but the measure is also directly integrated into the users' workflow (not necessarily visible to the user, but tracked by the system). Both semantic enhancements to process management and the capabilities of the adaptive process management system, which enables the dynamic change of running workflow instances, are used. Details can be found in [25].

Example 6 (Measure Selection): *To enable automated support for quality measures, the Company introduced the facilities for automated problem and quality opportunity detection. A GQM plan was created with maintainability and reliability as well as the creation of new functionality as goals. If developers now finish early on Assignments, the system can automatically assign them quality measures that fit the project goals and are appropriate to the personal situation. As example for this, consider refactoring of complex code. This measure was triggered as problems in the source code were detected, for example by applying the cyclomatic complexity metric. The measure was prioritized as high since it was judged as important and applicable to both the maintainability and reliability goals of the project. Finally, the system can choose the matching person for the measure based on properties like the skills of the person, their familiarity with that code section, or the amount of time they can spend in accomplishing a measure vs. the expected time needed for the measure.*

I. Quality Trend Analysis

The continuous monitoring of the quality of the source code is essential to be able to detect any impact of applied quality measures. Therefore, the list of quality measures from the *Rule Processing* module is utilized by the *Context Management* module. The list not only contains proposed measures, but also the metric belonging to each measure and the value of the metric. To enable automated evaluation of quality trends in conjunction with the GQM technique, different levels of key performance indicators (KPIs) were introduced as depicted in Figure 9.

KPIs are composite metrics unifying the values of other metrics or KPIs to enhance their expressiveness and significance. Each KPI is based on a formula that prescribes how the values of the encapsulated metrics are used to compute the KPI value. KPIs are utilized not only for quality trend analysis on different levels of abstraction, but also for automated goal deviation monitoring with respect to the GQM technique. Therefore, three levels of KPIs were introduced: on the most concrete level, the KPI unifies one or more metrics for clarity since different metrics may be utilized by the system. The QKPI represents a Question of the GQM technique as a value to facilitate automated deviation calculation, which is automatically computed from attributed KPIs and base metrics. The same applies for the GKPI, which unifies the values of the questions belonging to one project goal.

Compared to our initial approach described in [1] and [25], the calculation of the KPIs has been refined and moved

to the *Context Management* module. Therefore, the KPI structure and the various calculated KPI values are stored in the ontology for better information processing and access. The values can then be incorporated in reasoning procedures and the data can be easily provided to other external applications.

The calculations are now done in a uniform way for all KPIs, applying a weighted average of all values a *KPI* aggregates as depicted in Formula (1), where M_i are the concrete values that are aggregated and W_i are the attributed weights of the n metrics or KPIs being aggregated. All received metric values are normalized to a range from 0 to 1.

$$KPI = \frac{\sum_{i=1}^{i=n} W_i M_i}{\sum_{i=1}^{i=n} W_i} \quad (1)$$

J. Measure Assessment

Regarding different companies with different people, tools, and processes, applied measures may show different degrees of effectiveness. To reflect that and to improve future measure proposals, a so-called measure utility is introduced to indicate the usefulness of the applied measure. That property is neutrally initialized and updated after each application of the measure. The delta of the *KPI* related to the measure right after its application is compared to the value prior. Since some measures may not have an immediate effect, multiple future deltas can be taken into account. This process is further described in [25].

Example 7 (Measure Assessment): *The special refactoring proposed in Example 6 can now be automatically assessed for the Company since it has applied continuous quality measurement. If the refactoring is successful and the complexity of the code is reduced, this is indicated by a subsequent measurement showing a lower value for the cyclomatic complexity metric. This value will then also affect the value of a KPI related to the maintainability goal defined by the Company. The KPI value, in turn, will affect the utility value of the proposed refactoring measure, which will, if successfully applied, have a higher probability of selection by Context Management in the future.*

IV. Realization

This section provides implementation details for the components described in Section III and reflects their current implementation status.

A. Architecture

The technical architecture is depicted in Figure 10, whereby the modules are deployed as web services. To support loose coupling of the deployed services in CoSEEEK, event-driven computing and space-based computing are leveraged for service interaction [21]. The communication with the tuple space is technically realized using the web service framework Apache CXF.

For *SE Tools* and *Artifacts*, the actual instances are dependent on the SE environment and the current configuration. In the context of this paper, *SE Tools* includes static analysis tools like PMD, version control tools like Subversion, and IDEs (Integrated Development Environments) like Eclipse. Other relevant tools are, for example, external project management tools from which processes can be imported, like microTOOL inStep [32].

The primary *Artifacts* relevant to the scenario presented in this paper are source or test code files that are processed using these tools.

The *Event Extraction* module utilizes the Hackystat framework [38]. This framework provides a rich set of sensors that can be integrated into various SE tools. The sensors enable the *Event Extraction* module to automatically generate events in different situations, as, e.g., checking in a source code file in Subversion or switching to the debug perspective in Eclipse.

Most of the extracted events are rather atomic and thus combined by the *Event Processing* module to provide more semantic value. This is done utilizing Esper [39] for CEP. Esper provides a facility to define patterns that govern how certain events are combined to derive other higher-level events, which are then again written to the *Data Storage* module as all other events.

The *Data Storage* module, in turn, is realized via an implementation of the tuple space paradigm [40] on top of the XML database eXist [41] for shared XML data, whereas the Hackystat SensorBase is used for high volume event data. The specific CoSEEEK tuple space implementation that uses eXist consists of multiple so-called collections that structure the stored information. Examples include 'Context Management' as well as 'Process Management'. Each CoSEEEK module can write tuple events in these collections or subscribe to be automatically informed about new events in a certain collection.

The *Rules Processing* module automatically processes static rules to assign certain quality measures to certain violated metrics utilizing JBoss Drools [42].

The *AGQM* module, which is in charge of strategic quality measure prioritization, employs a multi-agent system (MAS) with different behavior agents. It is implemented utilizing the FIPA- compliant [43] Jade framework [44].

The *Context Management* module employs semantic technology to enable high-level utilization of all project knowledge. Technology advantages include enhanced interoperability between different applications, extending reuse possibilities, and the option for advanced content consistency checking [45]. It also provides a vocabulary for the modeled entities including taxonomies and logical statements about the entities. Ontologies also provide the capability of reasoning about the contained data and inferring new facts. As an ontology language, OWL-DL (Web Ontology Language Description Logic) [46] is used due to its proliferation and standardization. For simple RDF [47] based queries to the ontology, SPARQL [48] is used. Operations that are more complex are executed using the reasoner Pellet [49]. Programmatic access via DAO objects to the ontology is provided by the Jena framework [50]. Thus, different semantic concepts can be created and manipulated as needed.

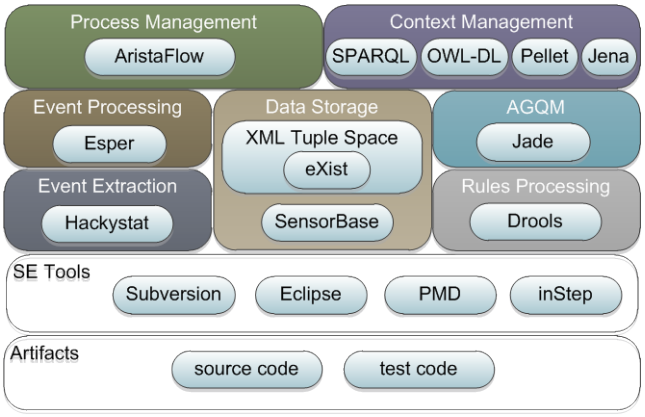

Figure 10. CoSEEEK Technical Architecture

The *Process Management* module builds upon the AristaFlow BPM Suite [51][52]. AristaFlow provides process management technology that is notable with respect to the flexible support of adaptive and dynamic workflows. New workflow templates can be composed out of existing application services in a plug-&-play like fashion, and then serve as schema for the robust and flexible execution of related workflow instances. In particular, during run-time, selected workflow instances can be dynamically and individually adapted in a correct and secure way; e.g., to deal with exceptional situations or evolving business needs [53]. Examples of workflow instance changes supported by AristaFlow include the dynamic insertion, deletion, or movement of single workflow activities or entire workflow fragments respectively (for a discussion of the adaptation patterns supported by AristaFlow, see [54]). For integrating these change functions and other AristaFlow services (e.g., for managing user work lists or for defining workflow templates) with domain- or application-specific process-aware information systems as in our case, the AristaFlow Open Application Program Interface (Open API) can be utilized [55][56]. For example, for dynamically inserting activities at the workflow instance level, the application developer can make use of the following system functions provided by the AristaFlow Open API:

- Querying the activity repository for available activity templates,
- Marking those activities of the workflow instance after which the selected activity shall be inserted (i.e., after completing these activities the newly added one shall be enabled),
- Retrieving the set of activities selectable as “end” activities for this insertion,
- Marking the activity (or set of activities) which shall serve as end activity (activities),
- Performing (tentatively) the insertion based on this information,
- Checking the AristaFlow report on detected errors (e.g., missing values for input parameters), and
- Making the instance change persistent.

Note that dynamic workflow instance changes can be conducted at a high level of abstraction. In particular, all complexity relating to dynamic workflow instance changes (e.g., correct transformations of the workflow schema, correct mapping of activity parameters, state adaptations) are hidden to a large degree from end users and application developers respectively [57]. Furthermore, AristaFlow provides techniques for learning from past experiences and workflow instance adaptations, respectively, and for evolving workflow schemes accordingly [58][59][60].

B. Context-aware Business Process Management

As mentioned in Section III, CPM (Context-aware Process Management) is enabled by correspondingly modeling the workflows in the ontology. Figure 11 illustrates this with the 'Develop Solution Increment' workflow of the OpenUP process [61].

In traditional process management, as mentioned in Requirement R:Sepm2, some aspects of task management have not been sufficiently covered. On the one hand, coarse-grained user assignments that have been planned for the current iteration or phase are not explicitly included since they are too abstract. In CoSEEEK, this is done explicitly in the *Context Management* module, as illustrated in Figure 11, by the Assignment 'Develop Feature X'. The latter is connected to the *Work Unit Container* that, in turn, contains all *Work Units* representing the activities of the workflow. If human tasks shall be executed via those activities, they are implicit parts of these activities.

In CPM, modeling it is done more explicitly. *Work Unit*s representing activities are only used for workflow governance. If they shall imply human activities, they have to be connected to Assignment Activities. The latter are also connected to the Assignment, making the connection of the abstract assignment to the concretely executed activities more explicit. However, activities like 'Implement Solution', in turn, consist of a number of smaller tasks. These tasks are also explicitly modeled in CPM. Figure 11 shows the *Atomic Tasks* related to the 'Implement Solution' *Assignment Activity*. These tasks can be automatically detected by the sensors of CoSEEEK’s *Event Extraction* module. Thus, it is possible that the system is automatically aware of the completion of an activity through the detected completion of all related tasks and thus can automatically finish that activity and propose the next one. This relieves the user from the burden of always explicitly informing the system about activity completion. The same applies to the Assignment, which can be automatically finished by the completion of all related Assignment Activities.

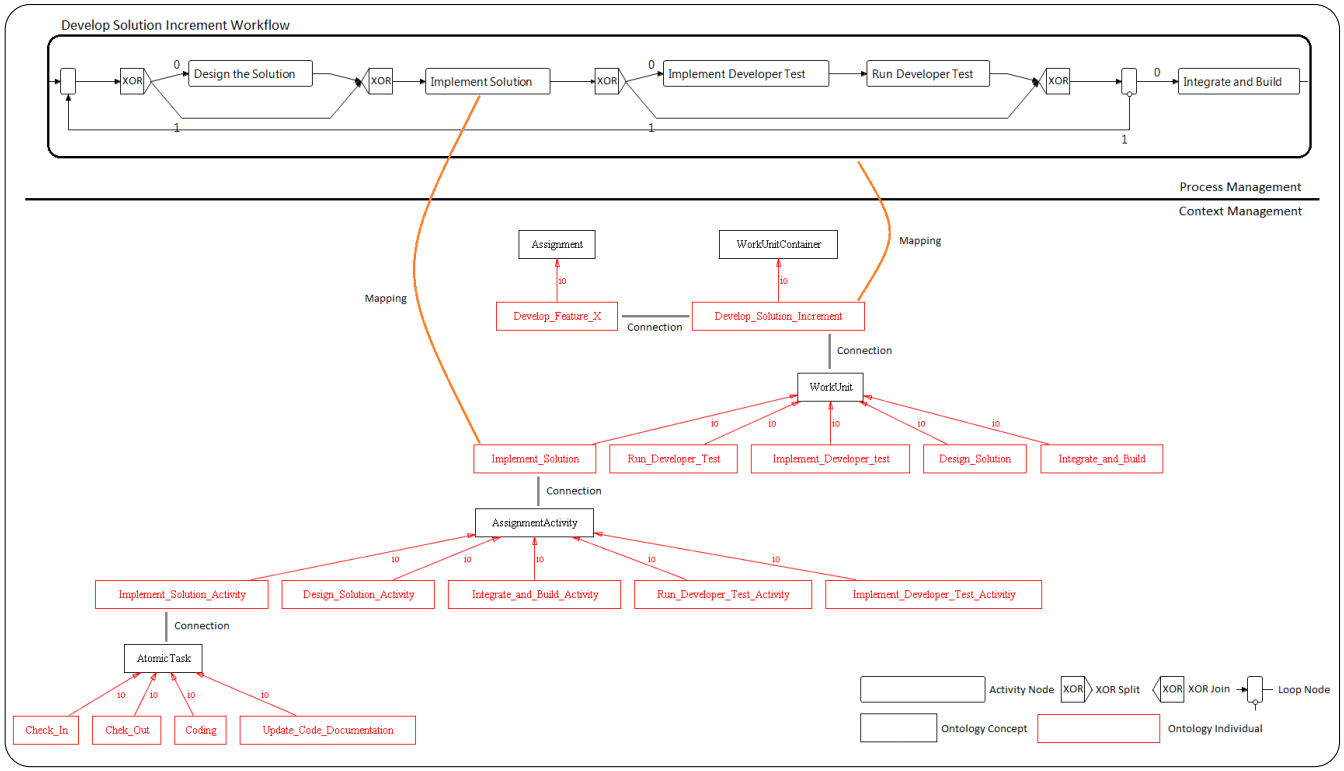

Figure 11. Concepts in Process Management and in the Ontology.

The enrichment of process management concepts via the ontology further enables the aforementioned extended vertical relations between workflows and the clear separation between *horizontal* and *vertical process management* (cf. Section III.C). The separation into two areas governed by two different modules was chosen despite the high coordination effort it imposes. The main reasons for this are as follows: mirroring process management components in the ontology not only enables extended vertical process management, but also allows for a tighter integration of the processes in the projects using different task levels for humans and CoSEEEK's environmental awareness capabilities. Since the decision on whether or not a *Work Unit* and the respective node can be completed is done in the *Context Management* module, the vertical dependencies are integrated into that procedure as well. Thus, multiple dependencies are all managed at one point, fostering contextual integration of process management and dependency extensibility. A *Work Unit* cannot complete, for instance, if related user activities are not completed or if the *Work Unit* depends on activities by other users or teams. An example of new dependencies that can be easily integrated is that an artifact has to be in a certain state or that an external tool has signaled a certain event. The horizontal governing of a process structure stays with process management because this is a non-trivial task and mature process management systems such as AristaFlow (cf. Section IV.A) support many correctness checks and guarantees on process execution. The extensions to *vertical process management* made by CPM are detailed in the following. There exist three possible connections, all of which can occur multiple times and can be mixed:

- *Depends-on-Work Unit Container*: This is the classical connection between an activity and a contained sub-workflow. When the *Work Unit* is activated, the *Sub Work Unit Container* is started and the *Work Unit* must not complete until the *Sub Work Unit Container* is completed. This connection is illustrated in Figure 6 by the *Work Unit* 'B4' that depends on the *Work Unit Container* containing the *Work Units* 'C1' – 'C4'.
- *Depends-on-Work Unit*: In this connection, the completion of the current *Work Unit* does not depend on a *Work Unit Container,* but on the completion of another *Work Unit*. When the depending *Work Unit* is activated and the *Work Unit Container* containing the *Work Unit* on which the current *Work Unit* depends has not been running yet, it is started. This connection is illustrated by the *Work Units* 'A3' and 'B4' in Figure 6.
- *Initiates-Work Unit Container*: This connection is asynchronous. The *Work Unit* does not depend on anything, but when it becomes activated, the connected *Work Unit Container* is started.

C. *Procedure*

This section gives a short outline about the temporal coordination of different modules. The whole procedure can be decomposed into three processes partly dependent on each other:

- When a report from an analysis tool is received, the tool-specific format is first transformed into a

unified one. Thereafter, the report is processed by the *Rule Processing* module with triggered measures output to *Data Storage* whereby any subscribers are notified. They retrieve and use the data, in this case the reactive section of the *AGQM* module and the KPI calculation of the *Context Management* module.

- When a user finishes an activity, the Q-Slot detection is started within the *Context Management* module. If a Q-Slot is available, the *AGQM* module is triggered by the *Context Management* module to generate an ordered list of proposed measures. This list is used by the tailoring process in *Context Management* to select a measure that is then integrated into a workflow by the *Process Management* module.
- At certain configured time points, the *Context Management* module is triggered to do an assessment of the applied measures. This relies on the applied measures and the calculated KPIs.

D. Problem Detection

Problem detection relies on the receipt of information about CoSEEEK's environment. This is indicated by events in the CoSEEEK infrastructure. For reports of external tools, these events include the location of the reports. Other facts like the execution of load or functional tests may only be indicated by events and are continuously monitored. When reports are received, the *Rule Processing* module is triggered. To be able to automatically evaluate metrics and to assign appropriate measures if thresholds are exceeded, the module must be aware of the metrics, the measures, the thresholds, and the tool used to measure the metrics. Thus, we developed a GUI to more easily define the involved items as depicted in Figure 12.

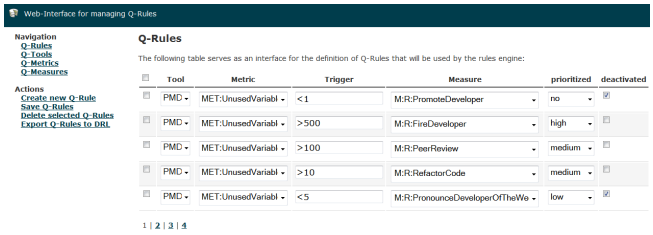

Figure 12. Rules GUI.

The rules produced by this GUI have the structure defined in Listing 1 and allow for the definition of rule priorities. The latter are used if, for example, two rules with different thresholds have been defined for the same metric and only one should be executed if both are triggered. The rules are then exported, transformed into the DRL format utilized by JBoss Drools, and loaded by the *Rules Processing* service. Any new reports then utilize the new rule set.

Listing 1: Rule example

```
<rule
 ID="12"
 Tool="PMD"
 Metric="MET:UnnecessaryConstructor"
 Trigger="&gt;=1"
 Measure="M:R:PeerReview"
 Prioritized="2"
/>
```

Rule processing produces unified XML reports containing all metrics whose thresholds have been violated and their associated triggered quality measures.

E. Measure Proposal

The output of a new unified report triggers the *Context Management* module to start the measure selection. For the prioritization of the measures, the *AGQM* module is triggered. This subsection gives some details about the agents utilized in the *AGQM* module.

The agent structure is defined as depicted in Figure 13. The AGQM agent is responsible for managing the agent module. It instantiates the other agents and determines whether a reactive or proactive measure will be proposed. For each defined goal, one goal agent is instantiated. In the proactive section, the goal agents communicate with the so-called session agent to realize the bidding process. The session agent takes the role of the "buyer" and thus selects the proactive measure from the goal agent with the highest bid. Each goal agent places bids according to its strategy. For the initial implementation, basic strategies were used. The three strategies 'offensive', 'balanced', and 'defensive' influence the starting bid of the agents as well as win-or-lose adaptation based on the last session. The strategy pattern allows these algorithms to vary. If insufficient points are left for the intended bid, the agent bids all points he has left. If an agent has no points left, it cannot place bids anymore until all agents have no points left, whereupon all points are reset to their initial value. Each agent has a list of proactive measures it could offer. Goals that are known to be at risk due to GKPI deviation are elevated to participation status in the bidding. If no report containing GKPI violations is received, all agents participate.

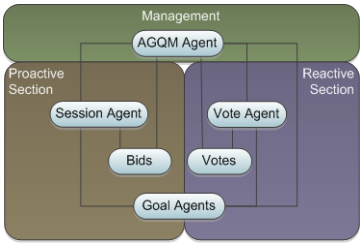

Figure 13. Agent Structure.

The reactive section is realized via the vote agent. Each time a report is received, the vote agent creates a weighted list of reactive measures using the report. To elicit the weight of each measure, the vote agent communicates with the goal agents. For each measure, a goal agent evaluates whether that measure is associated to its goal via the aforementioned

connection of measures, metrics, KPIs, and goals. In each voting process, a goal agent distributes all of its assigned points (initially allocated at the beginning of the iteration) uniformly across all measures in the current report that are associated to its goal. If multiple agents vote on one measure, the points are aggregated. If no report has been received yet, the voting process cannot be conducted. In that case, a proactive session is substituted.

That way the *AGQM* module creates a new ordered list of measures that mirror the predefined importance of the project's quality goals.

F. Measure Application

This part of the process was discussed in our initial approach [25]. However, due to technical issues, it has been extended and refined and this subsection presents how the integration of new quality measure activities into the user's workflow is accomplished. Therefore, new items have to be inserted both on the *Context Management* side and the *Process Management* side. This is done at first in the *Context Management* module. A quality measure is inserted as a new *Assignment* comprising certain *AssignmentActivities* and a separate *WorkUnitContainer* with *WorkUnits* and potential new *ExtensionPoints*. These are created from pre-specified template concepts that are connected to the *MeasureTemplate* (that is mentioned in Figure 17). To be able to insert the quality measure at the specified point, a new *WorkUnit* is created and inserted there, which is then connected to the newly created *WorkUnitContainer* belonging to the quality measure. This is illustrated in Figure 14.

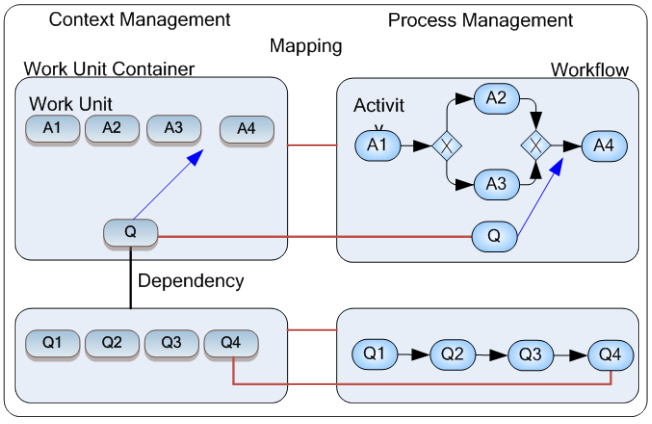

Figure 14. Measure Integration.

However, the insertion itself has to be also done within process management and therefore takes place later. In order to adapt a running workflow instance in AristaFlow, it has to be suspended from execution to apply the adaptations. This cannot be done if an activity in the instance is still active. However, at the time the quality measure integration is triggered, the activity that caused the *Q-Slot* generation is still active. Therefore, suspension is delayed until the activity is completed. This procedure is shown in Figure 15.

After the new individuals (*WorkUnits*, etc.) in the ontology have been created, a so-called *DeferredAction* is created and assigned to the *ExtensionPoint* where the measure should be integrated. That action will be automatically executed when the *WorkUnit* related to the *ExtensionPoint* is finished and will integrate the *Activity* containing the measure (named 'Q' in Figure 14) in the workflow instance. After that, a 'soft suspend' event is sent to *Process Management*, causing AristaFlow to do a soft suspend on the respective workflow instance. Thus, the instance will be automatically suspended right after the currently running activity is finished. This happens when the related *Work Unit* finishes via a 'signal Activity' event. This, in turn, causes the final integration of the quality measure activity in *Process Management* and is mirrored in *Context Management*.

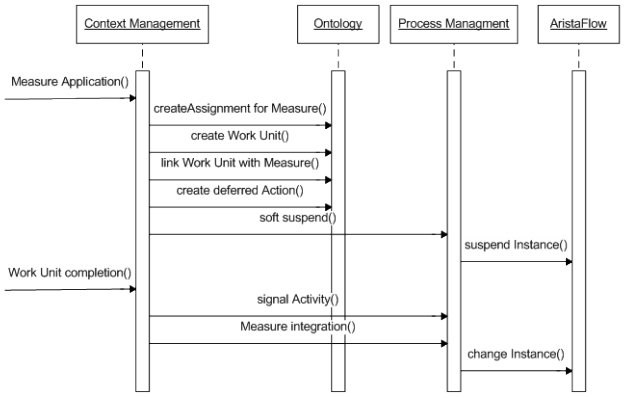

Figure 15. Measure Integration Procedure.

The order of the activities in the integration procedure was chosen in a way that the current activity is still running while the measure integration process starts. This was done to allow the insertion of a quality measure directly after every activity even if this is the activity the user currently processes that caused the creation of a *Q-Slot*. The soft suspend immediately suspends the workflow instance after activity completion and therefore no other activity is started. Then, the quality measure activity can be integrated and executed as the next activity if appropriate.

G. Quality Trend Analysis

The quality trend analysis is conducted in the *Context Management* module and the values are stored in the ontology. The concepts are illustrated in Figure 16.

Both *Metric* and *KPI* are united under the concept of the *QualityIndicator*. All concepts are separated into a template for the definition and a form containing a concrete value. When a *ViolationList* containing multiple *Metrics* is received, it is determined which *KPI* can be calculated via the *KPITemplate* and the *MetricTemplate*. For the computed values, new *KPIs* are then created.

To be able to do a uniform calculation, all received metric values are normalized to values between 0 and 1 where 1 is the best possible value and 0 is the worst possible one. Therefore, as part of the *MetricTemplate*, there is a defined maximum saved. The actual value is divided through this maximum to derive a value between 0 and 1. It also

defines a limit for the value, e.g., if a maximum of 15 has been defined as maximum for cyclomatic complexity, this would be the worst possible value. If the actual values had exceeded this limit, then 15 would be taken instead. There is also a property called *negative*, which indicates whether high values are negative (bad) or positive (good) indicators.

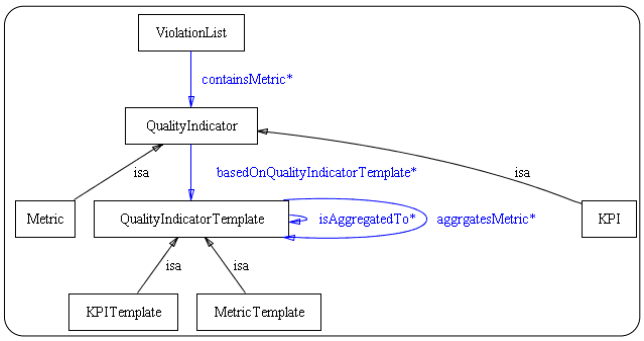

Figure 16. Quality Trend Analysis.

If a metric value is not available, the calculation will be done without that value. For some metrics, the absence of a value is also a negative indicator and thus a standard value can be defined in the *MetricTemplate*. If, for example, a metric had indicated the degree of functional testing compliance (a measure for the outcome of functional testing), its absence would indicate that no functional testing has been done yet. Since that fact should not be overlooked, a standard value can be defined.

It is also possible to integrate values from external tools as KPIs. If this is the case, a property 'external' can be used to indicate this. The *KPI* calculation is a weighted average and therefore each *KPITemplate* stores the weight used for that *KPI*.

H. Measure Assessment

In this part of the process, the calculated values of the KPIs are utilized for recalculating the measure utility factor. This is done using the changes (deltas) of the KPI values. For now, up to ten such deltas starting from the time of the measure application can be used. The concepts in the ontology realizing this are depicted in Figure 17.

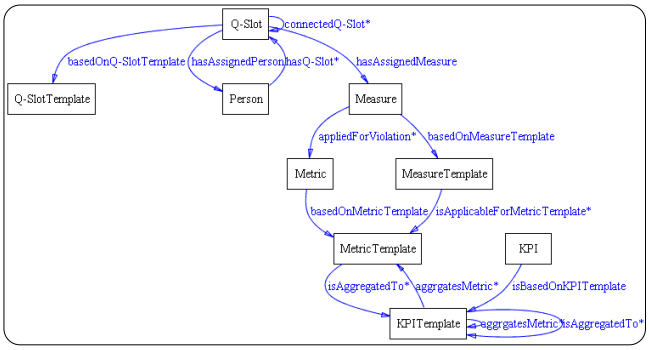

Figure 17. Measure Assessment.

More details on this calculation are provided in [25]. Compared to our initial approach, the ontology structure has been refined featuring the separation into template concepts and concrete ones for all involved concepts. Thus, they conform to the overall structure, implying a strict separation of the definition of certain items and their concrete values. That way, additional plausibility checks are also possible like the check whether a measure is permitted for a certain metric violation.

I. Modeling Effort

The presented approach implies modeling workflows and a myriad of extensions to them in the ontology. This leads to a relatively high modeling effort. That effort can nevertheless be limited: When the modeling is done utilizing the SE workflow language we developed in [24], both the concepts for the *Process Management* and the *Context Management* modules are automatically generated. If workflows are already in place in a workflow management system, the basic concepts in the ontology (*Work Units* and *Work Unit Co*ntainer) can be automatically generated and then be annotated manually. This could also limit the effort required for migrating to CoSEEEK in a company. If a supported workflow management system is in place there, only the annotations (e.g., *Extension Points*) have to be added manually. CoSEEEK will also include predefined metric sets and associated measures, standard SE processes, and GQM plans to facilitate the introduction of the system. That way small and medium sized companies could easily benefit from the higher level of automation CoSEEEK provides.

V. Evaluation

A scenario was constructed and technical measurements taken to evaluate the overall feasibility of the approach.

A. Scenario

Due to the large number of configuration factors involved and the breadth and depth of the approach we developed, a controlled scenario-based evaluation that combines real results with synthetic facts was chosen for initial feasibility testing. As input for code analysis, the org.eclipse.osee.framework.database package of the open source Eclipse Open System Engineering Environment was used.

1) Process

As a software development process, OpenUP [61] was chosen, a simplified free derivative of the Unified Process [62]. This process constitutes an iterative process featuring four project phases. In the *Inception phase* the scope of the project is defined, the use cases are outlined, risks are identified, and candidate architectures are selected. The *Elaboration phase* serves for capturing a healthy majority of system requirements and for addressing known risk factors. In addition, the system architecture is established and validated. In the *Construction phase* the system features are built based on the selected architecture. In the *Transition phase*, the system is deployed to the users. Each phase contains a number of iterations to complete its goals. Figure 18 shows how the OpenUP process can be used in the given scenario.

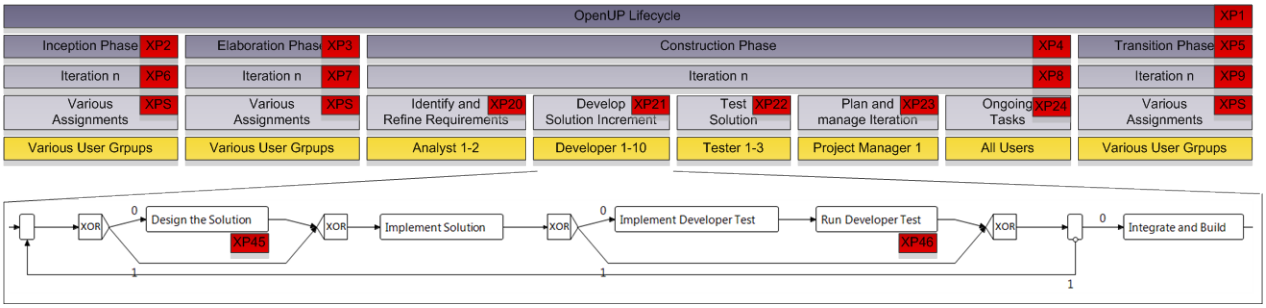

Figure 18. OpenUP Process with Configured Extension Points

Since the Construction phase is the largest phase in the project comprising most development activities, it was the focus of our evaluation. Figure 18 shows the other phases in a compressed way. The focus here is on the developers' activities, thus the 'Develop Solution Increment' workflow is shown in detail in the figure. Overall, 54 *ExtensionPoints* (XPs) have been defined for the workflows as depicted in the figure. The ones that are relevant for the developers' activities in a Construction iteration are XP8 at the end of the iteration, XP21 for measures to be applied between two assignments, and XP45 and XP46 for measures directly relating to coding or testing regarding certain artifacts.

2) GQM Plan

For the test scenario, a GQM plan was created to enable the AGQM agent processes shown in Table I. Four goals have been chosen: maintainability, reliability, performance, and functionality. This is just a simplified example of what is possible and can be incorporated and tailored by a quality manager.

The different metrics and KPIs that are part of the plan are illustrated in Appendix A. To measure the reliability of the code, different kinds of metrics have been chosen. On the one hand, well-known source code metrics like McCabe's cyclomatic complexity [63] or Nejmeh's npath complexity [64] have been used. On the other hand, metric suites were integrated, namely Chidamber and Kemerer's metrics suite [65] as well as the QMOOD metrics suite [66]. According to a study conducted in [67], these are good predictors for fault proneness and thus for reliability. Another factor that could affect the reliability of source code is whether it is covered by unit tests. This metric can be provided by tools like Cobertura [36] or EMMA [37] (see [35] for a comparison). Since, via sensors, it is possible to detect the execution of various tools for various activities, other factors can be used as metrics as well. An example for this is the degree of load testing that can also be an indicator of (the lack of) code reliability confidence.

For maintainability, a set of source code metrics have been selected and grouped to a question concerning the understandability of the code. To enhance the prediction quality of the goal, KPI external approaches have also been integrated: the maintainability index (MI) [68][69][70] is a formula proven to be a good predictor of maintainability and can be provided by the tool jhawk [71]. Maintainability can be also affected by certain problems in the source code called code smells. These can be detected via the DECOR approach [72], which is taken into account as well.

TABLE I. EXAMPLE GQM PLAN.

GKPI	QKPI	KPI	Metric
GKPI:REL	QKPI:CK		MET:WMC MET:DIT MET:NOC MET:CBO MET:RFC MET:LCOM
	QKPI:QMOOD		MET:ANA MET:CAM MET:CIS MET:DAM MET:DCC MET:MOA MET:MFA MET:NOM
	QKPI:COMP		MET:CYC MET:NPA
	QKPI:DD		MET:DD
	QKPI:CC		MET:CC
	QKPI:DLT		MET:DLT
GKPI:MAINT	QKPI:UND	KPI:CSV	MET:CR MET:TMM MET:UEM MET:UEC MET:ECB MET:TMF
	QKPI:CC		MET:CC
	QKPI:CSD		MET:DECOR
	QKPI:MI		MET:JHAWK
GKPI:FUNC	QKPI:UCC		MET:UCC
	QKPI:FTCF		MET:FTCF
GKPI:PERF	QKPI:CTAF		MET:CTAF
	QKPI:PRAF		MET:PRAF
	QKPI:PTCF		MET:PTCF

The implementation of all desired functionality is covered by the functionality goal. Thus, two metrics have been chosen to measure that. The use case coverage indicates how much of the desired functionality is implemented. The *functional testing compliance factor*, in turn, indicates how

many of the functional tests were passed. If no functional testing has been performed yet, the value of the functional testing compliance factor will be 0 in the worst case.

The performance goal comprises a metric called the *performance testing compliance factor*, which is similar to the *functional testing compliance factor*, but deals with performance tests. The other two metrics are related to the code optimization activities of code tuning and profiling.

3) Concrete situation

The scenario is targeted for a construction iteration of the OpenUP process that takes two weeks implying ten workdays. Ten developers are assumed to be part of the team and each developer has ten 'Develop Solution Increment' assignments, which are assumed to take one day each. Each night reports from code analysis tools are received as part of the nightly build process. As static analysis tool, PMD [33] is used; Appendix B shows the results of a report concerning the selected OSEE module. These results also include the threshold for each metric defined via the Rule module. For the concrete iteration, the focus is improving the quality of the source code, especially maintainability, since the functionality metrics are not violated, meaning the desired functionality is largely implemented. Therefore, the goal agents have been defined as depicted in Table II.

TABLE II. GOAL AGENT CONFIGURATION.

Agent	Points	Strategy
MAINT	100	Offensive
REL	80	Balanced
PERF	80	Balanced
FUNC	60	Defensive

For this scenario, the three strategies used for the agents have been defined as shown in Table III. As stated in Section III.F, they comprise three values: a start bid indicating how many of the distributed points an agent uses for its first bid and raise / reduce values indicating how the agent raises (reduces) its bid in case of loss (win).

TABLE III. AGENT STRATEGIES.

Strategy	Start bid	Raise	Reduce
Offensive	35%	20%	10%
Balanced	30%	15%	13%
Defensive	25%	10%	20%

Each time a *Q-Slot* occurs, the *AGQM* module is triggered to output an ordered list of proposed quality measures. For the current scenario, a 50:50 ratio between proactive and reactive measures was defined. Table IV shows the first ten proposed quality measures generated for a Q-Slot. Proactive measures are identified by the prefix "M:P:" and the assigned goal, reactive measures by "M:R:". The related metric whose threshold was violated for reactive measures is also shown.

TABLE IV. PROPOSED QUALITY MEASURES FROM AGQM

Slot	Quality Measure	Related Metric	ID
1	M:P:MAINT:Analyze Reuse Possibilities		m1
2	M:R::Increase Code Coverage	MET:CC	m2
3	M:R.Refactor Code	MET:ECB	m3
4	M:P:MAINT:Review Style Guidelines		m4
5	M:P:REL:Analyze Error Handling Implementation		m5
6	M:R:MAINT:Refactor Code	MET:TMM	m6
7	M:P:PERF:Do Profiling		m7
8	M:P:MAINT:Analyze Modularity		m8
9	M:R:PERF:Do Performance Testing	MET:PTCF	m9
10	M:R:Refactor Code	MET:CYC	m10

To determine the impact of the strategies in conjunction with the distribution of points in the proactive section, Table V shows the agents' bids for the slots, in which proactive measures were proposed. The numbers in parenthesis indicate the bid an agent would have placed according to its strategy when insufficient points were available.

TABLE V. AGENTS BIDS.

Slot	Winner	FUNC	REL	MAINT	PERF
1	MAINT	**35**	24	24	15
4	MAINT	**31**	28	28	17
5	REL	28	**32**	32	19
7	PERF	34	28	**37**	21
8	MAINT	**34(41)**	32	32	23

The results correlate with the expected arrangement of the proposed measures, where maintainability measures should be favored most, followed by reliability and performance measures.

For simplicity, in the current scenario, only early activity completion is assumed to have no defined quality overhead factor. Thus, the creation of Q-Slots only relies on execution time deviations of the assignments. These execution time deviations are shown in Table VI. Positive values indicate that an activity took less time than estimated, negative values indicate longer actual execution times, and grey boxes indicate *Assignments* after which the measure proposal process is started for the respective developer. For this scenario, it was assumed that a quality measure is possible if at least two hours are available.

With these values, five *Q-Slots* are possible in the iteration under consideration for the developers dev1, dev3, dev5, dev9, and dev10. For each Q-Slot, a measure from the list provided by the AGQM module has been selected, proposed, and assessed after application. The chosen measures, the applying developer, and the chosen *ExtensionPoints* are shown in Table VII. The measure utility has been initialized to '1' for all measures in the scenario. The table also shows the relating *KPI* used for assessment and the newly calculated 'measure utility' for the applied

measures. The calculations of the proactive measures have not been included here because in that limited scenario the GKPIs could not reflect an impact of the proactive measures. For a scenario with more details on measure tailoring and measure assessment, we refer to [25].

TABLE VI. EXECUTION TIME DEVIATIONS.

Developer	Assignment									
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
dev1	1	0	2	0	0	-1	1	1	1	1
dev2	0	1	-1	-1	0	1	0	1	1	1
dev3	1	0	-1	0	1	2	0	0	0	0
dev4	0	1	0	0	-2	2	0	-1	-1	-1
dev5	1	-1	1	0	2	0	0	0	0	0
dev6	-4	1	1	1	0	0	0	-1	-1	-1
dev7	1	0	0	0	-1	-2	1	1	1	1
dev8	0	1	0	1	0	1	0	1	0	0
dev9	0	1	0	0	0	0	2	0	0	0
dev10	1	0	-1	0	0	1	2	0	0	0

TABLE VII. APPLIED MEASURES.

Measure	Developer	Extension Point	KPI	Measure Utility
m1	dev1	21	GKPI:MAINT	1
m2	dev3	21	QKPI:CC	1.17
m3	dev5	21	KPI:CSV	1.17
m5	dev9	8	GKPI:REL	1
m9	dev10	21	QKPI:PTCF	1.29

While the scenario is not detailed and broad enough to ensure the applicability for the majority of SE real-world use cases, it shows the feasibility and potential of the approach towards addressing automated GQM and SQM. Future work will include trials of this approach with our industry project partners where empirical results can be evaluated.

B. *Performance Measurements*

For evaluating the technology and realization choices, performance measurements were conducted. Two different hardware configurations were utilized since the performance testing was performed by different developers on their own hardware (notebooks). Configuration A consisted of a computer with an AMD Turion II Dual-Core Mobile M500 2.2 GHz processor and 4 GB RAM. The software used was Windows 7 64-bit, Java Runtime Environment 1.6.0_16, Scala 2.7.7 final, Drools 5.1.0, Apache Ant 1.8.0, Apache CXF 2.2.4, and eXist 1.4.0. Configuration B consisted of one computer with an Intel Core i7 Q820 1.73 GHz processor and 6GB RAM. The software used was Windows 7 64-bit, the Java Runtime Environment 1.5.0_20, Apache CXF 2.2.4., eXist 1.2.6 (rev. 9165) and Jade 3.7. The tests were executed in a virtual machine (VMware Player 3.0.1 build-227600) assigned two processor cores and 4GB RAM.

All performance measurements were conducted five times consecutively, taking the average of the last three measurements. The first measurement series deals with the rule module and uses Configuration A and the second series covers the AGQM module and uses Configuration B. Other parts of the concept have been measured in [25].

1) *Rules processing*

Since the largest, most diverse, and regularly occurring amount of data to be analyzed by CoSEEEK are likely to be tool reports, and since the number of thresholds and quality measures needed to manage these can grow correspondingly, the scalability and performance of the *Rules Processing* was measured.

For the rule sets, both the loading latency and the execution time were measured for different numbers of rule sets as depicted in Table VIII. The XML report contained 4000 items generated to violate all of the rule sets that were defined for the test.

TABLE VIII. RULE PROCESSING PERFORMANCE.

Number of rules	Loading latency (sec)	Execution time (sec)
250	3.2	1.5
500	6.6	3.1
750	9.5	4.4
1000	13.0	5.8
1250	13.7	7.5
1500	16.3	8.6
1750	25.8	12.1

For scalability, the measurements show an almost linear increase of computation time. The loading performance is acceptable given that changes in rules, where reloading is necessary, should not be as frequent as rule execution. Since typical SE low-level activities are usually multiple minutes long, the execution time for the worst case measured (12 seconds) is still tolerable, making the approach suitable for the practical use in SE environments. Note that the rule engine would typically be run on a server and not on a notebook.

2) *AGQM*

For the AGQM module, two measurements were conducted to determine the impact of the number of goals (agents) and measures. First, the reactive measure list creation latency based on voting was measured. Second, the whole measure proposal process for a *Q-Slot* was measured.

The latency for vote list creation with varying numbers of measures and goals is depicted in Table IX. The results show that the number of measures has a greater impact on the latency than any increase in the number of goal agents voting (when measurement inaccuracies regarding the smaller values are disregarded).

TABLE IX. AVERAGE VOTE LIST CREATION LATENCY (MS) VS. GOALS AND MEASURES.

Measures	50	100	500	1000
5 Goals	111	194	273	924
10 Goals	113	160	815	1927
15 Goals	110	263	787	2090
50 Goals	92	317	842	2453
100 Goals	91	342	864	3003

A second measurement considered the measure proposal latency for a slot. It was assumed that for every goal exactly one proactive measure was defined, thus only the number of goals was of interest. All agents were given an offensive strategy and 100 points. For reactive measures, the measure list for voting was already prepared, from which only the first position was retrieved for simplification. The results are shown in Table X.

TABLE X. AVERAGE MEASURE PROPOSAL LATENCY (MS) VS. GOALS.

	5 Goals	10 Goals	15 Goals	50 Goals	100 Goals
Proactive	47	51	45	65	3211
Reactive	40	325	338	492	665

The reactive part shows the overhead of increasing agents for retrieving the top measure from the vote list. The proactive part remains constant for low goal numbers and then reaches an inflection point with a large number of goal agents. One possible explanation is extended bidding and thrashing with thread-based agents - this should be further investigated.

In summary, the performance of the current implementation appears to be sufficient for use in SEEs when the number of goals and measures used are within expected limitations. Performance could become an issue in large teams or projects or when large numbers of reactive measures are triggered. One way to address this would be to tune the *Rules Processing Module* to limit the number of reactive measures for which voting takes place. As to goal scalability, a large number of goals and goal agents would also imply a high degree of configuration overhead for a quality manager, thus likely naturally limiting the number of goals. Should nevertheless a large number of goals be desirable, distributing the agents could be considered.

VI. RELATED WORK

This section provides related work concerning our approach. It is structured into subsections covering the different topics of GQM support, contextual integration, and automated process adaptation.

A. GQM support

The combination of GQM with agents has been used for providing automated support for GQM plan creation [73][74][75] and for the computation of values for questions and goals [76][77]. In [75], a goal-driven use case method is utilized to elicit requirements. A set of agents assists the user in identifying goals and questions that are then used by another agent to obtain metrics. The collection of the measurement data and the creation of the measurement plan are then executed by two other agents. The ISMS (Intelligent Software Measurement System) [73][74] follows a similar approach using different groups of agents for user assistance and determination of different parts of the GQM plan. In [76][77], agents are used in the requirements process of the SW-CMM (Software Capability Maturity Model) model. The focus is the measurement and analysis of software processes using agents and fuzzy logic.

The approach presented in [78] aims at automated user assistance in GQM plan creation and execution but does not utilize agent technology. A tool was developed which allows creating GQM plans that use predefined forms as well as verifying the structural consistency of the plan and the reuse of its components. Furthermore, the tool supports data interpretation and analysis through aggregation of collected data. This approach is extended in [79], which integrates GQM more tightly with a development process to support GQM plan creation by an explicit process model.

For better integrating the GQM technique into the project flow via automation, different approaches were considered. [80] aims at integrating measurement programs as well as data collection into explicit process models, while [81] provides an object- oriented process model whose target is measurement. [82] proposes the usage of process models for creating GQM plans. Finally, the tool Prometheus [83] links executive plans with process models.

An approach extending the GQM technique is presented in [84]. It adds concepts such as entities, attributes, and units. cGQM [85] proposes the use of the Hackystat framework for GQM, applying continuous measurement with short feedback loops.

Other applications of agent technology include its utilization for automatic information retrieval [86], process monitoring [87], and collaboration support [88].

As opposed to the aforementioned approaches, CoSEEEK's AGQM process integrates its techniques into live software engineering environments, actively injecting SQM countermeasure proposals as guidance for developers. Agent technology is used differently in that the aim is neither user assistance in GQM plan creation nor assistance in interpreting measurement results. It is rather the fully automatic monitoring of goal fulfillment and the automatic assignment of quality measures for different types of quality deviations.

B. Contextual Integration of Process Management

Adapting application services to contextual changes is a major research area in areas like pervasive computing. A number of context-aware frameworks have been suggested to facilitate the implementation of application services that can somehow adapt their behavior to changing context. Frameworks like Context Management [89], CASS [90], SOCAM [91], and CORTEX [92] provide support for gathering and processing context data similar to our approach. However, they leave the reaction to context changes to the application or use hard-to-maintain rule-based approaches for dealing with respective changes.

Only few approaches like inContext [93] combine workflows with context-awareness as described in this paper. Regarding inContext, contextual information plays a central role similar to our approach; inContext strongly focuses on the teamwork domain, while our approach delivers a more generic technology enabling the development of context-aware, adaptive workflows.

The semantic annotation of process specifications to enable some method of contextual integration for the latter was addressed by various approaches. The focus of COBRA [94] is business process analysis. It presents a core ontology for business process analysis to provide better and easier analysis of processes to comply with standards or laws like the Sarbanes-Oxley act. A semantic business process repository is presented in [95]. It fosters automation of the business process lifecycle. It features capabilities for checking in and out as well as locking and options for simple querying and complex reasoning.

The approach presented in [96] aims at facilitating process models across various model representations and languages. This is achieved by multiple levels of semantic annotations: a meta-model annotation, a model content annotation, and a model profile annotation as well as a process template modeling language. [97] provides a concept for machine-readable process models to achieve better integration and automation. It utilizes a combination of Petri Nets and an ontology, whereas direct mappings of Petri Net concepts in the ontology are established. The approach described in [98] proposes an effective method for managing and evaluating business processes. This is realized via the combination of semantic and agent technology to monitor business processes. In contrast to the framework presented in this paper, these approaches do not consider the active intervention of a system in the execution of workflows. CoSEEEK exploits semantic annotation of the processes to a greater extent, using them to do context-based process adaptations.

C. *Automated Process Adaptation*

In the field of business process management, there exist several approaches supporting automated and dynamic adaptations of workflows during run-time [99]. As in our approach, their aim is to reduce error-prone and costly manual workflow adaptations during run-time and thus to relieve users from this task. As opposed to the presented work, the focus of these approaches is on automated exception handling. For this, the process-aware information system must be able to automatically detect exceptional situations, derive the dynamic change necessary to handle them, identify the workflows to be adapted, correctly apply the dynamic change to these workflows, and notify respective users. Existing approaches can be classified according to the basic method used for automatic exception detection and workflow adaptation:

Rule-based approaches. ECA-based (Event-Condition-Action) models are suggested for automatically detecting exceptional situations and determining the actions (i.e., workflow adaptations) required to handle them. In many ECA approaches, however, adaptations are restricted to currently enabled and running activities (e.g., to abort, redo, or skip activity execution) [100]. In contrast, AgentWork [101] further enables automated adaptations of the yet not entered regions of a running workflow (e.g., to add or delete activities). Basic to this is a temporal ECA rule model that allows specifying process adaptations at an abstract level and independent from a particular process model. When an ECA rule fires during run-time, temporal estimates are made to determine which parts of a running process instance are affected by the identified exception. These parts are then adapted immediately (predictive change) or, if this is not possible due to temporal uncertainty, at the time they are entered (reactive change).

Goal-based approaches formalize process goals (e.g., process outputs) and automatically derive the process model (i.e., the activities to be performed and their execution order) based on which these goals can be achieved. Further, if an exception (e.g., an activity failure) occurs during run-time that violates the formal goals, the process instance model is adapted accordingly. In ACT [102] for example, certain workflow adaptations (e.g., replacing a failed activity by an alternative one) are automatically performed if an activity failure leads to a goal violation. EPOS [103] rewrites software engineering workflows when process goals themselves change. Both approaches apply planning techniques to automatically derive and repair workflows in such cases. However, current planning methods do not cover all relevant process scenarios like our approach since important aspects (e.g., treatment of loops, appropriate handling of data flow) are not adequately considered.

Product-driven approaches interpret complex data structures representing a product in order to derive related workflow structures. Corepro, for example, allows product engineers to define complex data structures and to semi-automatically derive workflow structures from them [104]. The latter comprise the concrete workflows for engineering a particular product component (i.e., part) as well as the required synchronization between them. In particular, ad-hoc changes of a product structure are automatically compiled into respective adaptations of the workflow structure (on condition that certain correctness constraints are met). Corepro uses object life cycles and their dependencies in order to represent product components and their relations. DYNAMITE, in turn, uses graph grammars and graph reduction rules for defining the way in which a software engineering workflow may evolve over time [105]. Automatic adaptations are performed depending on the outcomes of previous activity executions (e.g., a design of a software module). Recently, more generic approaches aiming at a tighter integration or process and data have emerged (see [106][107] for an overview). These are particularly interesting for enabling artifact-based processes as in SE. For example, PHILharmonicFlows enables object-aware processes, which consider *object behavior* (i.e., the behavior of single objects and artifacts respectively) as well as *object interactions* (i.e., the coordinated processing of a collection of objects) [27]. Consequently, object-aware processes are based on two levels of granularity. In particular, data-driven process execution is enabled as well as integrated access to processes and data [108].

VII. CONCLUSION

SQA should be aligned to the SE process being used, and be relevant and applicable at the operation level. The manual combination of SQA with SEPM requires constant vigilance and associated labor in order to avoid missing quality

opportunities, to continuously monitor quality goal states, and to adapt measure and measure utility to new quality situations. The application of BPM in SE environments has been sparse due, among other factors, to a lack of contextual adaptability.

Automated quality guidance support could assist developers by providing SQA triggering that is based on current and factual data, continuously monitoring quality goal states and trends, and selecting and tailoring measure selection to that being most appropriate in the current situation. A set of requirements regarding context-awareness, process management, and quality measure selection was established in Section II.

Since the quality data and its analysis is not foreknown for reactive measures, and since there are limited time and resources for proactive measures, an automated selection of activity-based quality measures is beneficial. CoSEEEK's context-aware approach situationally adapts SE processes and ensures that quality opportunities are leveraged with the most appropriate measures for the current project quality risks. These are inserted into the appropriate point in the developer's workflow while taking developer properties such as competencies or available time into account. Quality risks can thus be mitigated and automation support can reduce inefficiencies.

Metric data from various tools can be integrated, thresholds can be continuously monitored, and appropriate measures can be triggered when the thresholds are exceeded. An automated awareness of schedule and early activity completion allows quality opportunities to be leveraged, and an overall quality overhead factor (could vary based on project phase) shall not to be exceeded. Process specification is extended to support flexible connections and dependencies between activities, enabling better context-based adaptations. GQM was extended for concrete metric-based automation support by agents. To deal with the expected plethora of reactive measures in projects, cooperative voting is used for reactive measure selection. For proactive measures, goals at risk bid against each other to allow importance and strategy to determine the point in time when proactive measures supporting their goals are proposed.

Measure selection is automatically tailored using a holistic project context comprising information about the project, tools, people, and their current situation. Measures are seamlessly integrated into running workflows using adaptive process management and semantic technology. Measure assessment adjusts the future use of measures based on their effectiveness, enabling the system to adjust and improve its SQA measure proposals.

A scenario-based evaluation exemplified the approach and showed its feasibility towards addressing automated GQM and SQM. Performance measurements indicated that the realization choices showed no significant scalability or performance issues.

Future work will assess the effectiveness of the approach via case studies in industrial settings. Concrete case studies at two companies have already been started and are expected to yield results soon. Work is also required to address the appropriate planning, determination, placement, and frequency of Q-slots in these industrial settings. More complex agent strategies, in addition to systematic detection of human expertise situations, will also be researched.

ACKNOWLEDGMENTS

The authors wish to acknowledge Andreas Kleiner, Stefan Lorenz, and Muhammer Tüfekci for their assistance with the implementation and evaluation. This work was sponsored by BMBF (Federal Ministry of Education and Research) of the Federal Republic of Germany under Contract No. 17N4809.

REFERENCES

[1] Grambow, G. and Oberhauser, R.: Towards Automated Context-Aware Selection of Software Quality Measures, Proc. of the 5th Int'l Conf. on Software Engineering Advances (ICSEA'10). IEEE Computer Society Press, 2010.

[2] Reijers, H.A. and van der Aalst, W.M.P.: The Effectiveness of Workflow Management Systems: Predictions and Lessons Learned. *Int'l Journal of Information Management*, 56(5), pp. 457-471, 2005.

[3] Heravizadeh, M.: Quality-aware Business Process Management. PhD thesis, Queensland University of Technology, Australia, 2009.

[4] Vollmer, K.: The EA View: BPM Has Become Mainstream, Forrester Research, 2008

[5] Mutschler, B., Reichert, M., and Bumiller, J.: Unleashing the Effectiveness of Process-oriented Information Systems: Problem Analysis, Critical Success Factors and Implications. IEEE Transactions on Systems, Man, and Cybernetics, 38(3), pp. 280-291, 2008.

[6] Brooks, F.P.: No Silver Bullet: Essence and Accidents of Software Engineering, Information Processing, 1986

[7] Glass, R.L.: Software Runaways: Monumental Software Disasters. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[8] Naur, P. and Randell, B.: Software engineering: Report of a conference sponsored by the NATO Science Committee. Garmisch, Germany, Scientific Affairs Division, NATO, 1968..

[9] Jones C.: Get Software Quality Right. In: Dr Dobb's Journal, June 28, 2010

[10] Eveleens, J.K. and Verhoef, C.: Quantifying IT forecast quality. Sci. Comput. Program. 74(11-12), pp. 934-88, 2009.

[11] Yourdon, E.: Death March, 2nd edition, Pearson Education, 2003

[12] Abdel-Hamid, T.: The economics of software quality assurance: a simulation-based case study, MIS Quarterly, 12(3), pp. 395-411, 1988.

[13] Kan, S.H.: Metrics and Models in Software Quality Engineering, Addison-Wesley, 2002

[14] Soini, J., Tenhunen, V., and Tukiainen, M.: Current Practices of Measuring Quality in Finnish Software Engineering Industry, In Richardson, I., Runeson, P., and Messnarz, R. (Eds.): Software Process Improvement, pp. 100-110, Springer, 2006.

[15] Gibson, D., Goldenson, D., and Kost, K.: Performance Results of CMMI-Based Process Improvement, Technical Report, CMU/SEI-2006-TR-004, Carnegie Mellon Software Engineering Institute, 2006

[16] McConnell, S.: Nine Deadly Sins of Project Planning, IEEE Software 18(5), pp. 5-7, 2001

[17] Slaughter, S.A., Harter, D.E., and Krishnan, M.S.: Evaluating the cost of software quality, Communications of the ACM, 41(8), pp. 67-73, 1998.

[18] Rausch, A., Bartelt, C., Ternité, T., and Kuhrmann, M.: The V-Modell XT Applied - Model-Driven and Document-Centric Development, Proc. 3rd World Congress for Software Quality, Vol. III, Online Supplement, pp. 131-138, 2005.

[19] WfMC. 1993. Workflow management coalition. http:// www. wfmc.org/

[20] Hill, J.B., Pezzini, M., and Natis, Y.V.: Findings: Confusion remains regarding BPM terminologies. Report No. G00155817, Gartner Research, 2008.

[21] Oberhauser, R.: Leveraging Semantic Web Computing for Context-Aware Software Engineering Environments, In: "Semantic Web, In-Tech, 2010.

[22] Grambow, G., Oberhauser, R., and Reichert, M.: Semantic Workflow Adaption in Support of Workflow Diversity, Proc. 4th Int'l Conf. on Advances in Semantic Processing (SEMAPRO'10), Florence, 2010, pp. 158-165

[23] Grambow, G., Oberhauser, R., and Reichert, M.: Semantically-Driven Workflow Generation using Declarative Modeling for Processes in Software Engineering, Proc. of the 4th Int'l Workshop on Evolutionary Business Processes, IEEE Computer Society Press (accepted for publication).

[24] Grambow, G., Oberhauser, R., and Reichert, M.: Towards a Software Engineering Workflow Language, Proc. 10th IASTED Conference on Software Engineering, Innsbruck, Austria, 2011.

[25] Grambow, G., Oberhauser, R., and Reichert, M.: Employing Semantically Driven Adaptation for Amalgamating Software Quality Assurance with Process Management, Proc 2nd Int'l Conf. on Adaptive and Self-adaptive Systems and Applications (ADAPTIVE'10), Lisbon, pp. 58-67, 2010.

[26] Müller, D., Herbst, J., Hammori, M., and Reichert, M.: IT Support for Release Management Processes in the Automotive Industry. Proc. 4th Int'l Conf. on Business Process Management (BPM'06), Vienna, Austria, pp. 368-377, 2006

[27] Künzle, V. and Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. Journal of Software Maintenance and Evolution: Research and Practice, 23(4), pp. 205-244, Wiley, 2011

[28] Künzle, V. and Reichert, M.: Integrating Users in Object-aware Process Management Systems: Issues and Challenges. Proc. BPM'09 Workshops, 5th Int'l Workshop on Business Process Design (BPD'09), Ulm, Germany, pp. 29-41, LNBIP 43(1), 2009.

[29] Sadiq, S., Orlowska, M., Sadiq, W., and Schulz, K.: When workows will not deliver: The case of contradicting work practice. In: Proc. BIS'05. (2005)

[30] Basili, V., Caldiera, G., and Rombach, H.D.: Goal Question Metric Approach, Encycl. of Software Engineering, John Wiley & Sons, pp. 528-532, 1994

[31] Luckham, D.C.: 'The power of events: an introduction to complex event processing in distributed enterprise systems' Addison-Wesley, 2001)

[32] microTOOL in-Step: http://www.microtool.de/instep/en/index.asp [Jan .2011]

[33] Copeland, T.: PMD Applied, Centennial Books, 2005

[34] Ayewah, N,, Hovemeyer, D., Morgenthaler, J. D., Penix, J., and Pugh, W.: Experiences using static analysis to find bugs, IEEE Software, 25(5), pp. 22-29, 2008.

[35] Yang, Q., Li, J.J., and Weiss, D.: A survey of coverage based testing tools, Proc. Intl. Workshop on Automation of Software Testing (AST'06), pp. 99–103. ACM Press, 2006.

[36] Cobertura http://www.cobertura.sourceforge.net [Jan 2011]

[37] EMMA http://www.emma.sourceforge.net [Jan 2011]

[38] Johnson, P. M.: Requirement and Design Trade-offs in Hackystat: An In-Process Software Engineering Measurement and Analysis System, Proc. of the 1st Int'l Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, pp. 81-90, 2007.

[39] Esper: http://esper.codehaus.org/ [Jan 2011]

[40] Gelernter, D.: Generative communication in Linda, ACM Transactions on Programming Languages and Systems, 7(1):80-112, 1985.

[41] Meier, W.: eXist: An Open Source Native XML Database, Web, Web-Services, and Database Systems, Springer, ,pp. 169-183, 2009.

[42] Browne, P.: JBoss Drools Business Rules. Packt P. Browne. JBoss Drools Business Rules. Packt Publishing, 2009.

[43] O'Brien, P.D. and Nicol, R.C.: FIPA — Towards a Standard for Software Agents, BT Technology Journal, 16 (3):51-59, 1998.

[44] Bellifemine, F., Poggi, A., and Rimassa, G.: JADE - A FIPA-compliant Agent Framework, Proc. 4th Int'l Conf. and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents. London, 1999.

[45] Gasevic, D., Djuric, D., and Devedzic, V.: Model driven Architecture and Ontology Development, Springer, 2006.

[46] World Wide Web Consortium: OWL Web Ontology Language Semantics and Abstract Syntax, 2004

[47] World Wide Web Consortium: Resource Description Framework (RDF) Concepts and Abstract Syntax, 2004

[48] Prud'hommeaux, E. and Seaborne, A.: 'SPARQL Query Language for RDF, W3C WD 4, 2006.

[49] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., and Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics 5(2), pp. 51-53, 2006

[50] McBride, B.: Jena: a semantic web toolkit, Internet Computing, 2002

[51] Dadam, P. and Reichert, M.: The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support - Challenges and Achievements. Computer Science - Research and Development, Springer. 23(2), pp. 81-97, 2009.

[52] Reichert, M. et al: Enabling Poka-Yoke Workflows with the AristaFlow BPM Suite. Proc. BPM'09 Demonstration Track, Ulm, Germany, 2009.

[53] Reichert, M., Rinderle-Ma, S., and Dadam, P.: Flexibility in Process-aware Information Systems. LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC), Special Issue on Concurrency in Process-aware Information Systems. LNCS 5460, pp. 115-135, 2009

[54] Weber, B., Reichert, M., and Rinderle-Ma, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. Data and Knowledge Engineering, Elsevier, 66(3), pp. 438-466, 2008

[55] Lanz, A., Kreher, U., Reichert, M., and Dadam, P.; Enabling Process Support for Advanced Applications with the AristaFlow BPM Suite. Proc. of the Business Process Management 2010 Demonstration Track, September 2010, Hoboken, New Jersey, USA.

[56] Reichert, M., Dadam, P., Rinderle-Ma, S., Jurisch, M., Kreher, U., and Goeser, K.: Architecural Principles and Components of Adaptive Process Management Technology. In: PRIMIUM - Process Innovation for Enterprise Software. Lecture Notes in Informatics , Vol. P-151, pp. 81-97, 2009.

[57] Reichert, M. and Dadam, P.: ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control. Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems, 10(2), pp. 93-129, 1998

[58] Li, C. and Reichert, M. and Wombacher, A.: Mining Business Process Variants: Challenges, Scenarios, Algorithms. Data & Knowledge Engineering, 70(5), pp. 409-434, Elsevier, 2011.

[59] Günther, C.W. and Rinderle-Ma, S. and Reichert, M. and van der Aalst, W.M.P. and Recker, J.: Using Process Mining to Learn from Process Changes in Evolutionary Systems. Int'l Journal of Business Process Integration and Management, Special Issue on Business Process Flexibility, 3(1), pp. 61-78, 2008.

[60] Weber, B. and Reichert, M. and Wild, W. and Rinderle-Ma, S.: Providing Integrated Life Cycle Support in Process-Aware Information Systems. Int'l Journal of Cooperative Information Systems, 18(1), pp. 115-165, World Scientific Publ, 2009.

[61] OpenUp http://epf.eclipse.org/wikis/openup/ [November 2010]

[62] Scott, K.: The Unified Process Explained, Addison-Wesley Longman Publishing Co., Inc., 2002

[63] T. J. McCabe: A complexity measure, IEEE Trans. Software Eng. 2(4), pp. 308-320, 1976.

[64] Nejmeh, B.A.: NPATH: A measure of execution path complexity and its applications. Comm. of the ACM, 31(2):188-200, 1988.

[65] Chidamber, S.R. and Kemerer, C.F.: A Metrics Suite for Object Oriented Design, IEEE Transactions on Software Engineering, 20 (6), pp. 476-493, 1994.

[66] Bansiya, J. and Davis, C.: A Hierarchical model for object-oriented design quality assessment, IEEE Transactions on Software Engineering, 28(1), pp. 4–17, 2002.

[67] Olague, H.M., Etzkorn, L.H., Gholston, S., and Quattlebaum, S.: Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes, IEEE Transactions Software Engineering., 33(6), pp. 402-419, 2007.

[68] Oman, P.W., Hagemeister, J., and Ash, D.: A Definition and Taxonomy for Software Maintainability, Technical Report #91-08-TR, Software Engineering Test Laboratory, University of Idaho, Moscow, ID, 1991.

[69] Coleman, D.: Assessing Maintainability, Proc. of the Software Engineering Productivity Conference 1992, Hewlett-Packard, Palo Alto, CA, pp. 525-532, 1992.

[70] Coleman, D., Ash, D., Lowther, B., and Oman, P.W.: Using Metrics to Evaluate Software System Maintainability, IEEE Computer, 27(8), pp. 44-49, 1994.

[71] JHawk http://www.virtualmachinery.com/jhawkprod.htm [November 2010]

[72] Moha, N., Gueheneuc, Y.G., Duchien, L., and Meur, A.F.: DECOR: A method for the specification and detection of code and design smells, IEEE Transactions on Software Engineering, 36(1), pp.:20-36, 2010.

[73] Chen, T., Homayoun Far, B., and Wang, Y.: Development of an Intelligent Agent-Based GQM Software Measurement System, Proc. 12th Asian Test Symposium (ATS), pp. 188-197, 2003.

[74] Junling Huang, Far, B.H.: Intelligent software measurement system (ISMS), Canadian Conf. on Electrical and Computer Engineering, pp. 1033–1036, 2005.

[75] FanJiang, Y.-Y and Wu, C.-H.: Towards a Multi-agents Architecture for GQM Measurement System, Proc. 9th Int'l Conf. on Hybrid Intelligent Systems, pp. 277-280, 2009.

[76] Seyyedi, M.A., Teshnehlab, M., and Shams, F.: Measuring software processes performance based on the fuzzy multi agent measurements, Proc. Intl Conf. on Information Technology: Coding and Computing - Volume II. ITCC. IEEE Computer Society, Washington, DC, pp. 410-415, 2005.

[77] Seyyedi, M.A., Shams, F., and Teshnehlab, M.: A New Method For Measuring Software Processes Within Software Capability Maturity Model Based On the Fuzzy Multi-Agent Measurements, Proc. World Academy Of Science, Engineering and Technology Vol. 4, pp. 257-262, 2005.

[78] Lavazza, L.: Providing automated support for the GQM measurement process, IEEE Software, 17(3), pp.:56-62, 2000.

[79] Lavazza, L. and Barresi, G.: Automated support for process-aware definition and execution of measurement plans, Proc. 27th Int'l Conf. on Software Engineering, pp. 234 – 243, 2005.

[80] Lott C.M. and Rombach H.D.: Measurement-based guidance of software projects using explicit project plans, Information and Software Technology 35(6-7), pp. 407-419, 1993.

[81] Morisio M.: Measurement Processes are Software Too, Journal of Systems and Software 49(1), pp. 17-31, 1999.

[82] Broeckers A., Differding C., and Threin G.: The Role of Software Process Modeling in Planning Industrial Measurement Programs, Proc. Int. Metrics Symposium, Berlin 1996.

[83] Visaggio, G.: Process Improvement Through Data Reuse, IEEE Software 11(4), pp. 76-85, 1994.

[84] De Panfilis, S., Kitchenham B., and Morfuni N.: Experiences introducing a measurement program, Information and Software Technology 39(11), pp. 745-754, 1997.

[85] Lofi C.: cGQM - Ein zielorientierter Ansatz für kontinuierliche, automatisierte Messzyklen, Proc. 4th National Conf. on Software Measurement and Metrics (DASMA MetriKon 2005), 2005.

[86] Pelletier, S.-J., Pierre, S., and Hoang, H.H.: Modeling a Multi-Agent System for Retrieving Information from Distributed Sources, Journal of Computing and Information Technology, 11(1), pp. 15-39, 2003.

[87] Wang, M., Wang, H., and Xu, D.: The design of intelligent workflow monitoring with agent technology, Knowledge-Based Systems, 18(6), pp. 257-266, 2005.

[88] Tan, W., Chen, R., Shen, W., Zhao, J., and Hao, Q.: An Agent-Based Collaborative Enterprise Modeling Environment Supporting Enterprise Process Evolution, Computer Supported Cooperative Work in Design III, pp. 217-226, 2007

[89] Korpipipää P. et al.: Managing context information in mobile devices. IEEE Pervasive Computing 2(3), pp.42-51, 2003

[90] Fahy, P. and Clarke, S.: CASS – a middleware for mobile context-aware applications. Proc. Workshop on Context-awareness (held in connection with MobiSys'04), 2004.

[91] Gu, T., Pung, H.K., and Zhang, D.Q.: A middleware for building context-aware mobile services. Proc. IEEE Vehicular Technology Conference (VTC), Milan, Italy, pp. 2656 – 2660, 2004.

[92] Biegel, G. and Cahill, V.: A framework for developing mobile, context-aware applications. Proc. 2nd IEEE Conference on Pervasive Computing and Communication, pp. 361 - 365 , 2004

[93] Dorn C., Dustdar S.: Sharing Hierarchical Context for Mobile Web services. Distributed and Parallel Databases 21(1), pp. 85-111, 2007.

[94] Pedrinaci, C., Domingue, J., and Alves de Medeiros, A.: A Core Ontology for Business Process Analysis, LNCS 5021, pp. 49-64, 2008.

[95] Ma, Z., Wetzstein, B., Anicic, D., Heymans, S., and Leymann, F.: Semantic Business Process Repository, Proc. Workshop on Semantic Business Process and Product Lifecycle Management, pp. 92–100, 2007

[96] Lin, Y. and Strasunskas, D.: Ontology-based Semantic Annotation of Process Templates for Reuse, Proc.10th Int'l Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'05), 2005.

[97] Koschmider, A. and Oberweis, A.: Ontology based Business Process Description, Proc. CAiSE´05 Workshops, pp. 321-333, 2005.

[98] Thomas, M., Redmond, R., Yoon, V., and Singh, R.: A Semantic Approach to Monitor Business Process Performance, Communications of the ACM 48(12), pp. 55-59, 2005

[99] Weber, B. and Sadiq, S. and Reichert, M:. Beyond Rigidity - Dynamic Process Lifecycle Support: A Survey on Dynamic Changes in Process-aware Information Systems. Computer Science - Research and Development, 23(2), pp. 47-65, Springer, 2009.

[100] Casati, F., Ceri, S., Paraboschi, S., and Pozzi, G.: Specification and implementation of exceptions in workflow management systems. ACM TODS, 24(3), pp. 405–451, 1999.

[101] Müller, R., Greiner, U., and Rahm, E.: AGENTWORK: A workflow system supporting rule–based workflow adaptation. Data & Knowledge Engineering, 51(2), pp. 223–256, 2004.

[102] Beckstein, C. and Klausner, J.: A planning framework for workflow management. Proc. Workshop Intelligent Workflow and Process Management. Stockholm, 1999.

[103] Liu, C. and Conradi, R.: Automatic replanning of task networks for process model evolution. Proc. European Software Engineering Conference, pp. 434–450. Garmisch, Germany, 1993.

[104] Müller, D., Reichert, M., and Herbst, J.: A new paradigm for the enactment and dynamic adaptation of data-driven process structures. Proc. CAiSE'08, pp. 48–63, 2008

[105] Heimann, P., Joeris, G., Krapp, C., and Westfechtel, B.: DYNAMITE: Dynamic task nets for software process management. Proc. Int'l Conf. Software Engineering (ICSE'06), pp. 331–341, 1996

[106] Künzle, V. and Weber, B. and Reichert, M.: Object-aware Business Processes: Fundamental Requirements and their Support in Existing Approaches. Int'l Journal of Information System Modeling and Design (IJISMD), 2(2), pp. 19-46, IGI Global, 2011.

[107] Künzle, V. and Reichert, M.: Striving for Object-aware Process Support: How Existing Approaches Fit Together In: Proc. 1st Int'l Symposium on Data-driven Process Discovery and Analysis, Campione d'Italia, 2011 (accepted for publication).

[108] Künzle, V. and Reichert, M.: A Modeling Paradigm for Integrating Processes and Data at the Micro Level. In: Proc. 12th Int'l Working Conference on Business Process Modeling, Development and Support (BPMDS'11), London, June 2011, LNBIP 81, pp. 201-215, 2011

APPENDIX A: LIST OF UTILIZED METRICS

GKPI:REL:Reliability
GKPI:MAINT:Maintainablity
GKPI:FUNC:Functionality
GKPI:PERF:Performance
QKPI:CK:ChidamberAndKemerer
QKPI:QMOOD:QmoodMetricsSuite
QKPI:COMP:Complexity
QKPI:DD:DefectDensity
QKPI:CC:CodeCoverage
QKPI:DLT:DegreeOfLoadTesting
QKPI:UND:Understandability
QKPI:CSD:CodeSmellDensity
QKPI:MI:MaintainablitiyIndex
QKPI:UCC:UseCaseCovrage
QKPI:FTCF:FunctionalTestingComplienaceFactor
QKPI:CTAF:CodeTuningActivityFactor
QKPI:PAF:ProfilingActivityFactor
QKPI:PTCF:PerformanceTestComplianceFactor
KPI:CSV:CodingStyleViolations
MET:WMC: Weighted Methods per Class
MET:DIT: Depth of Inheritance Tree
MET:NOC: Number of Childeren
MET:CBO: Coupling between Objects
MET:RFC: Response for Class
MET:LCOM: Lack of Cohesion in Methods
MET:ANA:AvgNumberOfAncestors
MET:CAM:CohesionAmongMethods
MET:CIS:ClassInterfaceSize
MET:DAM:DataAccessMetric
MET:DCC:DirectClassCoupling
MET:MOA:MeasureOfAggregation
MET:MFA:MeasureOfFunctionalAbstraction
MET:NOM:NumberOfMethods
MET:CYC: CyclomaticComplexity
MET:NPC:NPathComplexity
MET:DD: DefectDensity
MET:CC: CodeCoverage
MET:DLT: DegreeOfLoadTesting
MET:CR:CommentRatio
MET:TMM:TooManyMethods
MET:UEM:UncommentedEmptyMethod
MET:UEC:UncommentedEmptyConstructor
MET:ECB:EmptyCatchBlock
MET:TMF:TooManyFields
MET:UCC:UCC:UseCaseCovrage
MET:FTCF:FunctionalTestingComplienaceFactor
MET:CTAF:CodeTuningActivityFactor
MET:PAF:ProfilingActivityFactor
MET:PTCF:PerformanceTestComplianceFactor

APPENDIX B: PMD RESULTS

Metric	Value	Violation Threshold
MET:AccClGen	1	5
MET:AvoidDeeplyNestedIfStmts	2	5
MET:AvoidInstanceof ChecksInCatchClause	1	5
MET:AvoidReassigningParameters	1	5
MET:AvoidSynchronized AtMethodLevel	2	5
MET:ClassWithOnlyPrivate ConstructorsShouldBeFinal	4	10
MET:CloseResource	1	5
MET:CollapsibleIfStatements	1	20
MET:CompareObjectsWithEquals	1	5
MET:ConfusingTernary	6	20
MET:CyclomaticComplexity	4	2
MET:EmptyCatchBlock	2	1
MET:EmptyMethodInAbstract ClassShouldBeAbstract	2	5
MET:ExcessiveImports	1	5
MET:ExcessivePublicCount	1	5
MET:LooseCoupling	4	20
MET:NPathComplexity	1	5
MET:OverrideBothEquals AndHashcode	1	5
MET:PositionLiterals FirstInComparisons	1	5
MET:SimplifyBooleanExpressions	2	5
MET:SingularField	1	5
MET:StaticMethods	1	5
MET:SwitchStmts ShouldHaveDefault	1	5
MET:TooManyFields	1	5
MET:TooManyMethods	4	3
MET:Uncommented EmptyConstructor	5	5
MET:UncommentedEmptyMethod	5	5
MET:UnconditionalIfStatement	1	5
MET:UseCollectionIsEmpty	2	5
MET:UseLocaleWith CaseConversions	2	5